

Course Exercises Guide

Developing Applications in IBM Business Process Manager Advanced V8.5.7 - II

Course code WB861 / ZB861 ERC 1.2



March 2017 edition

Notices

This information was developed for products and services offered in the US.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
United States of America*

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

© Copyright International Business Machines Corporation 2017.

This document may not be reproduced in whole or in part without the prior written permission of IBM.

US Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Trademarksvi
User IDs and passwordsvii
.....	viii
Exercise 1. Using iterative development to create applications	1-1
1.1. Reviewing the exercise scenario	1-3
Part 1: Path one: Ineligible applications	1-4
Part 2: Path two: Eligible applications with LOW credit risk	1-12
Part 3: Path three: Eligible applications with HIGH credit risk	1-20
Part 4: Path four: Eligible applications with MED credit risk	1-36
1.2. Create a business process diagram in IBM Process Designer.....	1-49
Part 1: Associate a library with the repository	1-49
Part 2: Create a business process diagram	1-60
Part 3: Create an implementation for a business process activity	1-66
Part 4: Test the business process diagram by using Playback and Inspector	1-70
Part 5: Import a business process diagram into IBM Integration Designer	1-77
Part 6: Republish business process diagram import in IBM Integration Designer	1-79
Exercise 2. Version control for SCA applications	2-1
Introduction	2
Part 1: Version the FoundationModule and FoundationLibrary	2-2
Part 2: Deploy FoundationModule with the <i>serviceDeploy</i> tool	2-11
Part 3: Install and start the generated EAR file	2-13
Part 4: Test the versioned application with cross-component tracing	2-17
Part 5: Troubleshoot applications in the administration console and Failed Event Manager	2-25
Part 6: Create a second version of the module and library	2-39
Part 7: Use the <i>serviceDeploy</i> tool to publish the new version of FoundationModule	2-42
Part 8: Test the versioned application	2-46
Part 9: Clean up the environment	2-50
End of exercise52
Exercise 3. Working with SCA bindings and qualifiers	3-1
Implementing a WebSphere MQ bound import component	3
Part 1: Adding the WebSphere MQ import	3-3
Part 2: Updating the AccountVerification process component	3-8
Part 3: Testing the module	3-11
Part 4: Reviewing the test results	3-18
Part 5: Cleaning up the workspace	3-23
Using the event sequence quality of service qualifier25
Part 1: Adding the flat file adapter	3-25
Part 2: Testing without event sequencing	3-34
Part 3: Enabling the event sequencing qualifier	3-38
Part 4: Testing with event sequencing enabled	3-42
Part 5: Cleaning up the workspace	3-44
Exercise 4. Applying fault handlers	4-1
Fault handling3
Part 1: Creating a FaultMessage business object definition	4-3
Part 2: Adding an interface definition in the FoundationLibrary	4-4
Part 3: Create a CustomerType BPEL business process	4-6

Part 4: Implementing the CustomerType business process logic	4-12
Part 5: Integrating CustomerType with the AccountVerification process	4-18
Part 6: Adding a Fault Handler to AccountVerification	4-22
Part 7: Returning a FaultMessage	4-28
Part 8: Define the correlation set and property for AccountVerification	4-31
Part 9: Updating the FoundationModule assembly diagram	4-37
Part 10: Creating test data to test fault handler	4-38
Part 11: Testing fault handler in the AccountVerification process	4-39
Exercise 5. Applying a compensation handler to WS-BPEL	5-1
Compensation handling	3
Part 1: Extending the default fault handler in the AccountVerification process	5-3
Part 2: Implementing a new status validation in the AccountVerification process	5-7
Part 3: Adding a compensation handler	5-12
Part 4: Testing compensation handler in the AccountVerification process	5-12
Exercise 6. Working with business state machines	6-1
Introduction to the SodaMachine sample	2
Part 1: Explore the building blocks of the SodaMachine with the business state machine editor ..	6-3
Part 2: Test the SodaMachine state machine with BPEL Process Choreographer Explorer	6-20
End of exercise	32
Exercise 7. Defining transactional behavior in SCA applications	7-1
Part 1: View transaction propagation settings for a composite application by using an integration solution diagram	7-2
Part 2: View transaction propagation settings for an individual module by using the assembly diagram	7-6
Part 3: Examine the transaction boundaries for activities in long-running business processes	7-16
Part 4: Use a scope in a long-running business process to manage persistence	7-20
Exercise 8. Creating flexible business processes	8-1
Part 1: Adding a selector	8-2
Part 2: Testing the selector	8-31
Exploring process dynamicity	35
Exercise 9. Working with static relationships	9-1
Part 1: Create the lookup relationship	9-2
Part 2: Creating a data map to invoke the relationship	9-12
Part 3: Wiring the data map into the AccountVerification process	9-17
Part 4: Testing the application	9-20
Part 5: Reviewing the test results	9-28
Part 6: Reviewing the relationship data by using the Relationship Manager	9-31
Part 7: (Optional) Testing with an invalid country code	9-37
Exercise 10. Implementing a mediation flow	10-1
Part 1: Reviewing the workspace	10-2
Part 2: Creating the required business object for the aggregation mediation flow	10-3
Part 3: Implementing the mediation flow	10-5
Part 4: Testing the solution	10-27
Part 5: Cleaning up the workspace	10-33
Exercise 11. Writing a generic error handler for IBM Process Server	11-1
Part 1: Implement the generic error handler subflow	11-2
Part 2: Implement the generic subflow into a mediation flow	11-8
Part 3: Test the mediation flow with ErrorHandlingSubflow	11-13

Part 4: Clean up the environment	11-25
Exercise 12. Implementing security	12-1
Securing SCA applications2
Part 1: Preparing the workspace	12-2
Part 2: Creating user groups and users	12-2
Part 3: Implementing security on the AccountVerification process module	12-5
Part 4: Enabling IBM Process Server security	12-9
Part 5: Testing security	12-16
Securing activities of a process application in Process Designer.....	24
Part 1: Start the Process Center environment	12-24
Part 2: Creating users and groups	12-24
Part 3: Assigning a participant group to the lanes	12-28
Part 4: Test the business process diagram by using Playback and Inspector	12-33
Exercise 13. Applying governance to process applications	13-1
Introduction to the governance process for the snapshot installation2
Part 1: Explore the sample governance process application	13-2
Part 2: Enabling governance for a process application	13-13
Part 3: Testing the governance that is applied to the process application	13-16
Exercise 14. Integrating other applications with IBM Integration Designer	14-1
Integrating IBM Business Process Manager with IBM Business Monitor.....	.2
Part 1: Create a monitor model	14-2
Part 2: Import and explore a Business Monitor model	14-10
Part 3: Associate a monitor model with a process application	14-15

Trademarks

The reader should recognize that the following terms, which appear in the content of this training document, are official trademarks of IBM or other companies:

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide.

The following are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide:

CICS®	ClearCase®	DB™
DB2®	developerWorks®	FileNet®
IA®	IBM MobileFirst™	IMS™
PartnerWorld®	Passport Advantage®	Rational®
Redbooks®	Sametime®	WebSphere®

Intel and Intel Core are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

VMware and the VMware “boxes” logo and design, Virtual SMP and VMotion are registered trademarks or trademarks (the “Marks”) of VMware, Inc. in the United States and/or other jurisdictions.

Evolution® is a trademark or registered trademark of Kenexa, an IBM Company.

Other product and service names might be trademarks of IBM or other companies.

Exercises description

This course includes the following exercises:

- Exercise 1. Using iterative development to create applications
- Exercise 2. Version control for SCA applications
- Exercise 3. Working with SCA bindings and qualifiers
- Exercise 4. Applying fault handlers
- Exercise 5. Applying a compensation handler to WS-BPEL
- Exercise 6. Working with business state machines
- Exercise 7. Defining transactional behavior in SCA applications
- Exercise 8. Creating flexible business processes
- Exercise 9. Working with static relationships
- Exercise 10. Implementing a mediation flow
- Exercise 11. Writing a generic service handler
- Exercise 12. Implementing security
- Exercise 13. Applying governance to process applications
- Exercise 14. Integrating other applications with IBM Integration Designer

In the exercise instructions, you see that a line prefixes each step. You might want to check off each step as you complete it to track your progress.

Most exercises include required sections, which must always be completed. These sections might be required before you do later exercises. Some exercises might also include optional sections that you might want to do if you have sufficient time and want an extra challenge.

How to follow the exercise instructions

Exercise structure

Each exercise is divided into sections with a series of steps and substeps. The step represents an action to be performed. If required, the substeps provide guidance on completing the action.

1+1=2 Example

-
- 1. Create a user account named **ADMIN**.
 - a. Right-click **My Computer** and choose **Manage** from the menu.
 - b. Expand **Local Users and Groups**.
- ...continue*
-

In this example, the creation of a user account is the action to be performed. The substeps underneath provide specific guidance on how to create a user account. (In this example, the instructions are for Windows operating system.) Words that are highlighted in bold represent menu items, field names, and other interface elements.

An underscore precedes each step and substep. You are encouraged to use these markers to track your progress. As you complete a step, place an X or a check mark on the underscore to indicate that it is completed. By tracking your progress in this manner, you can stay focused when you experience interruptions during a lengthy exercise.

User IDs and passwords

The following table contains a list of user ID and password information for this course.

Entry point	User ID	Password
VMware image	Administrator	passw0rd
Windows 2012 R2	Administrator	passw0rd
Process Center Console	pcdeadmin	web1sphere
IBM Process Designer	pcdeadmin	web1sphere
Administration console for IBM Process Manager Advanced	bpmadmin	web1sphere
Administration console for Process Server test environment	admin	web1sphere



Important

Online course material updates might exist for this course. To check for updates, see the Instructor wiki at: <http://ibm.biz/CloudEduCourses>

Exercise 1. Using iterative development to create applications

Estimated time

02:00

Overview

In this exercise, you explore the iterative, model-driven development process. You create a business model in IBM Process Designer, take a snapshot of the application, and synchronize with IBM Integration Designer. You then add implementation details, wire the model artifacts into a business process, and test the business process in IBM Integration Designer.

Objectives

After completing this exercise, you should be able to:

- Examine the various paths for the Account Verification process in IBM Integration Designer
- Create a business process diagram in IBM Process Designer
- Build an implementation for an activity in a business process diagram
- Use the IBM Process Center Playback Server to test the business process diagram
- Replace a Java component with a business process diagram
- Use IBM Integration Designer to test the imported business process diagram

Introduction

IBM Process Designer is a tool that is used for modeling and developing simple business processes in IBM Business Process Manager. IBM Process Designer is examined in great detail in separate IBM Training courses. However, as demonstrated in this lab exercise, great business value can be derived by allowing business analysts with little development expertise to use IBM Process Designer to create business processes for your integration applications. The results of their efforts can then be maintained in the IBM Process Center repository for use in IBM Integration Designer. Technical experts provide the implementation artifacts for deployment of the automated solution.



Important

The exercises in this course use a set of lab files that might include scripts, applications, files, solution files, PI files, and others. The course lab files can be found in the following directory:

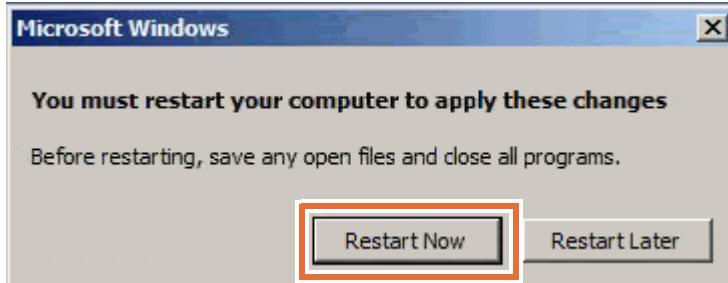
- C:\labfiles for the Windows platform
- /usr/labfiles for the Linux platform

The exercises point you to the lab files as you need them.



Note

Upon initial login, if the operating system displays a dialog box with the “You must restart your computer to apply these changes” message, click **Restart Now**.



Requirements

Completing the exercises for this course requires a lab environment. The environment includes the exercise support files, Mozilla Firefox, IBM Business Process Manager Advanced V8.5.7, IBM DB2 V10.1, RFHUTIL, WebSphere MQ V8.0, and the IBM Process Server V8.5.7 test environment.

Exercise instructions

1.1. Reviewing the exercise scenario

In this portion of the exercise, you are presented with an overview of the scenario that is used in the exercises. The scenario is taken from scenario that is built in course WB860: *Developing Applications for IBM Business Process Manager Advanced V8.5.7-1*. The purpose of this section is to review and familiarize yourself with the scenario that was developed in the previous course and is used as the basis for development in this course. Each section includes an overview of one path through the scenario with instructions for testing the path.

You can review this material to understand the mechanics of the applications that are used in the course. If you want to review how the applications operate, you might also look at this material in later exercises. As you proceed through the exercises, you add components to extend the functions of the scenario.

The “account verification” scenario that is used in the exercises is primarily composed of a business process that is designed to automate the processing of customer accounts. The purpose of this scenario is to process existing customer accounts by transferring and validating them through a new process. This scenario might be used in a real-world situation, when the data is moved from an old system to a system that includes IBM Business Process Manager.

The four basic paths through the process are:

1. WebSphere Adapter for Flat Files is used to archive the applications that are ineligible.
2. Eligible applications for customers that are deemed to be a low credit risk are automatically approved.
3. Eligible applications for customers that are deemed to be a high credit risk require supplemental documentation and manual approval or denial.
4. Eligible applications for customers that are deemed to be a medium credit risk also require supplemental documentation and manual approval or denial.

A predefined test case (with corresponding test data) is used to test each path through the process.

- An application for customer **AbcCo** is used to test the ineligible path (path 1 in the preceding paragraph).
- An application for customer **IBM** is used to test the low credit risk path (path 2 in the preceding paragraph).
- An application for customer **TestCo** is used to test the high credit risk path (path 3 in the preceding paragraph).
- An application for customer **ACME** is used to test the medium credit risk path (path 4 in the preceding paragraph).

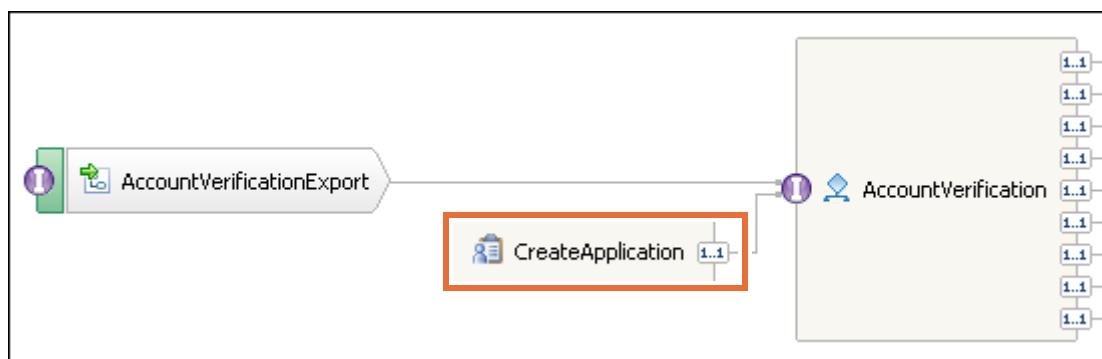
Part 1: Path one: Ineligible applications

Overview

In this path, the WebSphere Adapter for Flat Files is used to archive the applications that are ineligible. When you use the company name `AbcCo` to test the applications, the `eligibleApplication` attribute is set to `false`. When you use `companyName AbcCo` to submit an application, the application flows through these activities: **Account Verification Receive > Determine Application Eligibility > Map to Ineligible > Record Ineligible Application > Account Verification Reply**. Since the application is ineligible, the flat file adapter is used to archive the application to the file system for later processing or auditing.

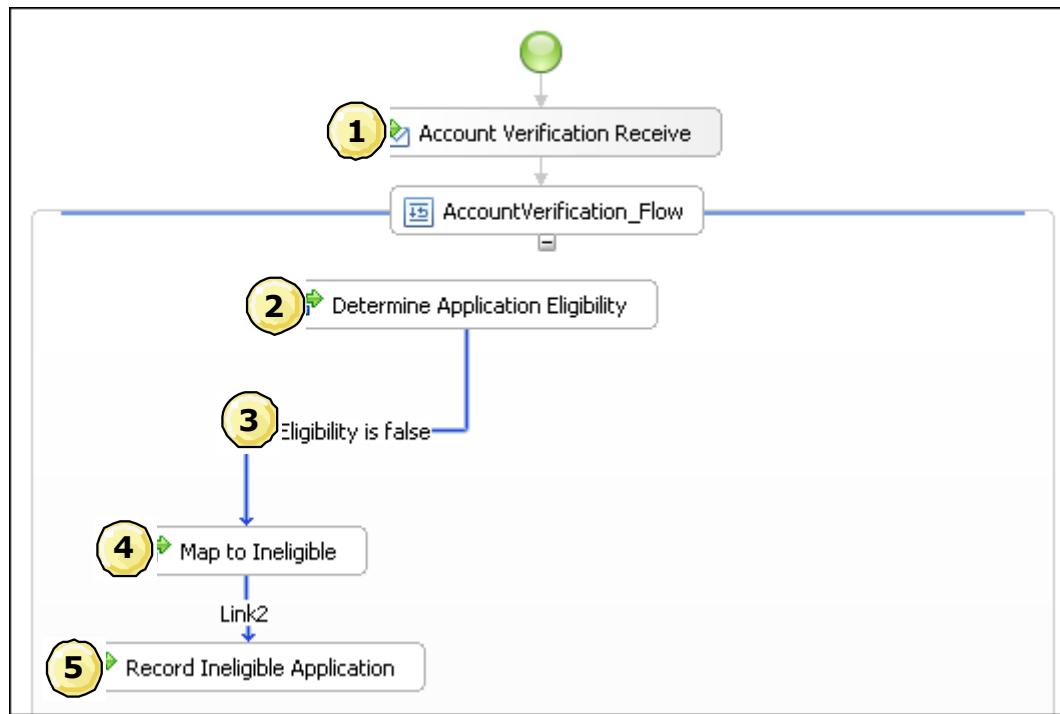
The first path through the process consists of the following steps:

1. `AccountOpeningUI`, a JSF user interface for the `CreateApplication` human task, is used to create an application and to trigger a new instance of the `AccountVerification` business process.



(Before the `CreateApplication` task is created, you use the IBM Integration Designer integrated test client to test your solution by calling the `AccountVerification` SCA component or the `AccountVerificationExport` component.)

2. When the application for **AbcCo** is received, it takes the ineligible path through the process. The ineligible path consists of the following activities:

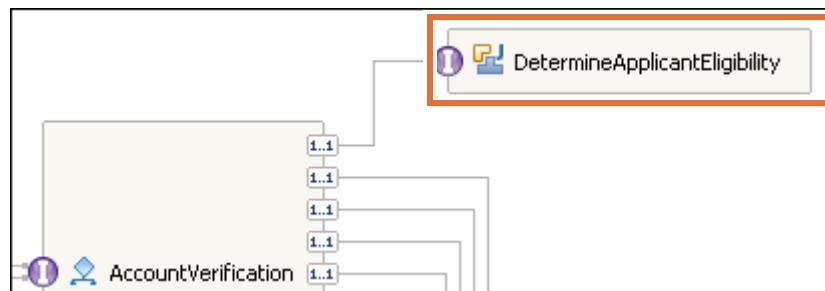


3. The Account Verification Receive activity receives the message (the application) that the CreateApplication invocation task sends.

The process follows the link to the next activity, the `AccountVerification_Flow` structured activity (a generalized flow).

4. The first activity in `AccountVerification_Flow` is the Determine Application Eligibility invoke activity. Determine Application Eligibility calls the Determine Applicant Eligibility service through the Determine Application Eligibility reference partner.

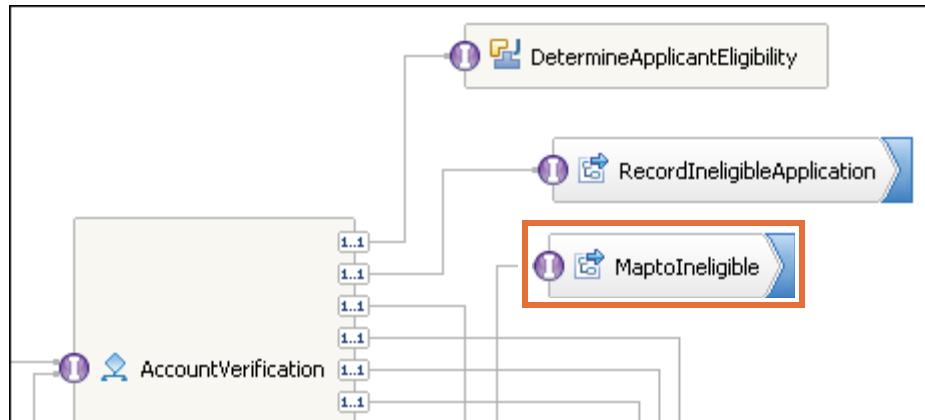
The Determine Applicant Eligibility service (in the `FoundationModule`) is an SCA component, with a Java implementation that uses simple Java code to return sample data for each of the four test cases.



By default, the Java code sets the `eligibleApplication` element in the `CustomerApplicationVariable` to `false` for **AbcCo** and to `true` for **ACME**, **IBM**, and **TestCo**. In practice, this element would not be set automatically.

5. The `eligibleApplication` element in `CustomerApplicationVariable` is examined. If `eligibleApplication` is `false` (which is always the case for **AbcCo**), then the “Eligibility is false” link is followed to the `Map to Ineligible` activity.
6. The `Map to Ineligible` activity uses the `MaptoIneligiblePartner` reference partner to call the services of the `IneligibleMediationService` module.

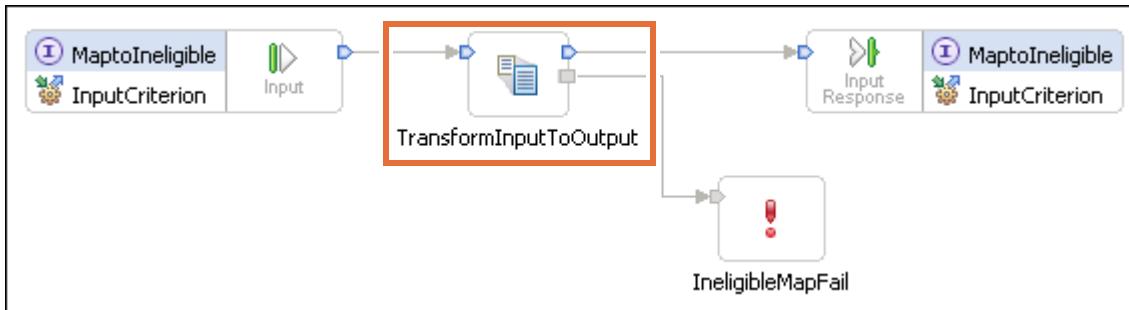
The `MaptoIneligiblePartner` reference partner calls the `MaptoIneligible` import component on the `FoundationModule` assembly diagram.



The `MaptoIneligible` import component calls the `IneligibleMediationExport` component in the `IneligibleMediationService` module.

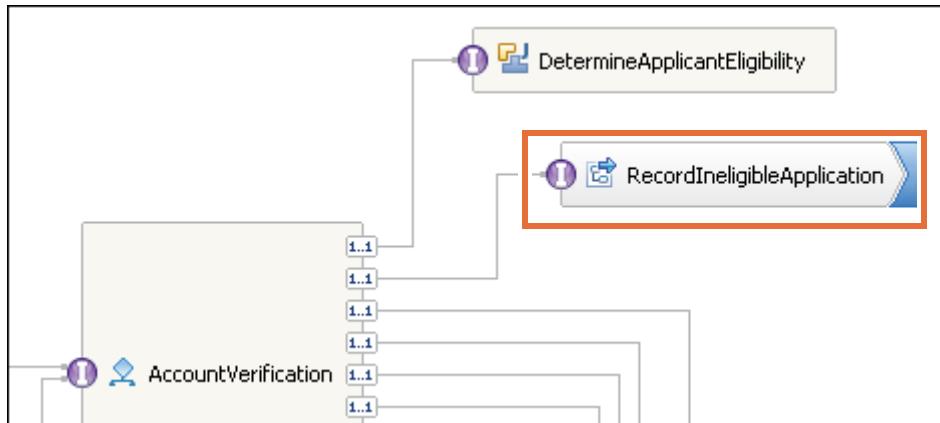


The `IneligibleMediation` mediation flow (accessible through the `IneligibleMediationExport` component) transforms the business object from a `CustomerApplication` into an `IneligibleApplication` business object. (The service that archives the application uses an `IneligibleApplication` input so a Mapping primitive is used to transform the data.)



7. When the data is transformed, the next link is followed to the `Record Ineligible Application` invoke activity. The `IneligibleMediationService` returns an `IneligibleApplication` business object. This business object is sent to the `Record Ineligible Application` service through the `RecordIneligibleApplication` reference partner.

The RecordIneligibleApplication reference partner calls the RecordIneligibleApplication import component on the FoundationModule assembly diagram.



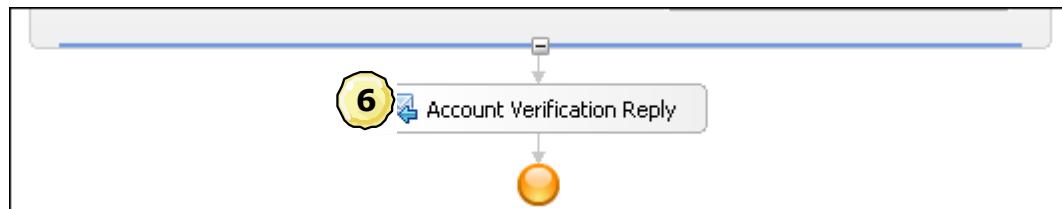
The RecordIneligibleApplication import component calls the RecordIneligibleApplicationExport component in the FoundationServices module.



In turn, the FlatFileOutboundImport component is called, which calls the flat file adapter code in the CWYFF_FlatFile module to write the IneligibleApplication to the file system.

8. RecordIneligibleApplication is the final activity in AccountVerification_Flow. The final link is followed from AccountVerification_Flow to the Account Verification Reply activity.

Account Verification Reply returns a Message business object to the client, which contains a message element that reads: "Account Verification recorded this application as ineligible for the customer: AbcCo." The RecordIneligibleApplication Java component on the FoundationServices assembly diagram sets this message text.



End-to-end test

Test an ineligible application:

- ___ 1. Open the scenario workspace.
- ___ a. On your desktop, open the folder that is labeled **Exercise Shortcuts**.
- ___ b. Double-click the shortcut that is labeled **Exercise 1**. IBM Integration Designer starts.
- ___ c. Wait for the workspace build to complete; the build might take a few minutes.

- ___ d. Close the **Getting Started** tab.
- ___ 2. Start the server and deploy all of the modules.
 - ___ a. In the **Servers** view, right-click **IBM Process Server v8.5.7 at localhost** and select **Start** from the menu. Wait for the start process to complete before you continue. The start might take several minutes. If the **Automatically Publish** window is displayed, click **OK**.

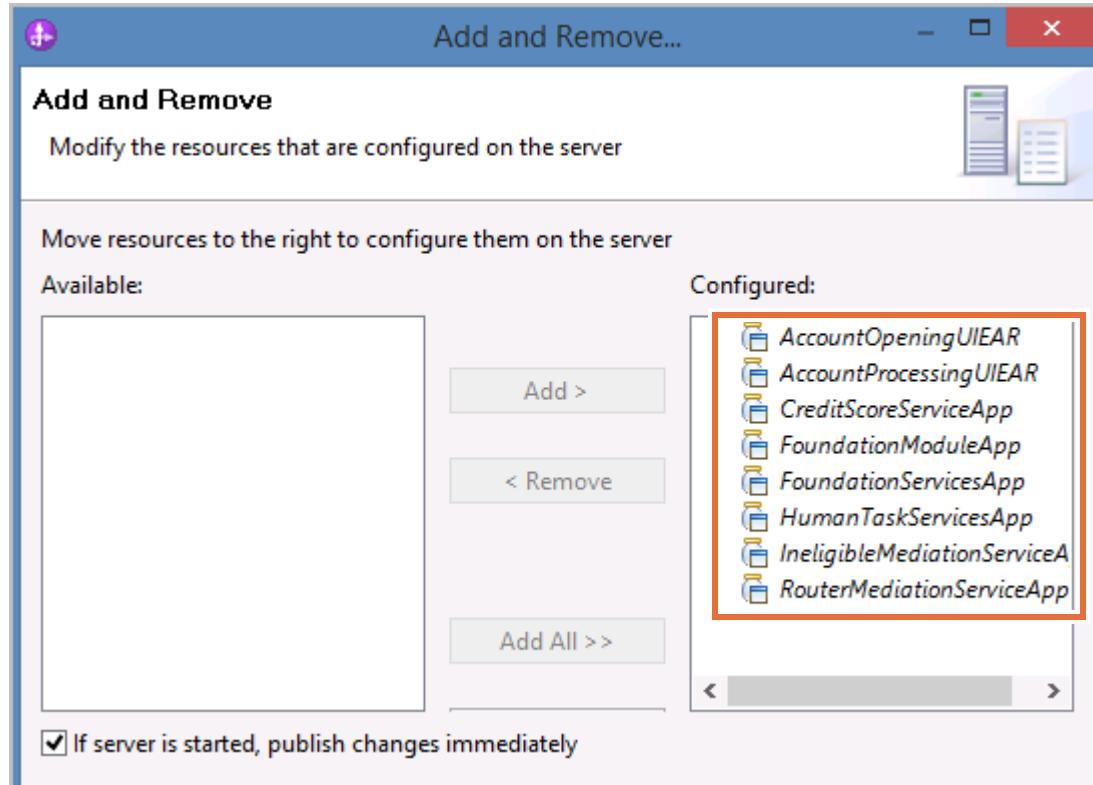
The server is started when the message **Server server1 open for e-business** is visible in the **Server Logs** view. The server status also changes to **[Started]** in the **Servers** view.



Note

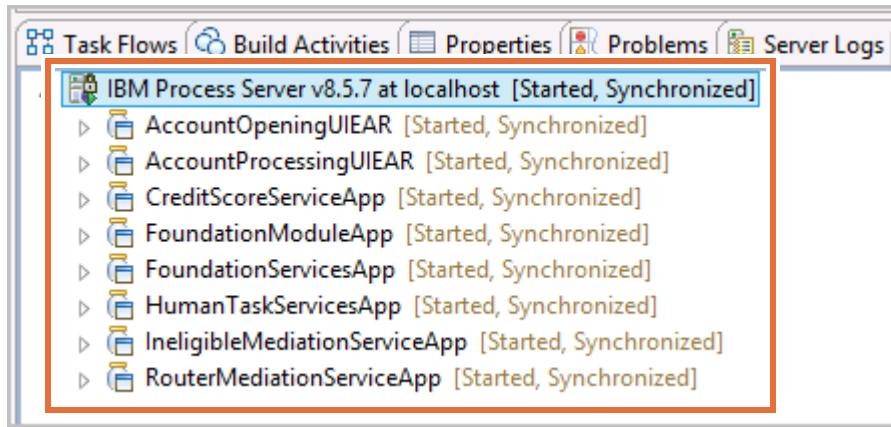
If the server start status shows for more than 5 minutes and the **Server Logs** view is not getting updated with server start messages, then you need to start it manually. Right-click **IBM Process Server v8.5.7 at localhost** and select **Stop** from the menu to stop successfully (do the Stop operation one more time if the server status does not show as Stopped). Restart the server by right-clicking **IBM Process Server v8.5.7 at localhost** and selecting **Start**.

- ___ b. In the **Servers** view, right-click **IBM Process Server v8.5.7 at localhost** and select **Add and Remove**.
- ___ c. Click **Add All** to add each of the projects to the **Configured** list.



- ___ d. Click **Finish**.

- __ e. Wait until the modules are published and started. The publish process might take few minutes, depending on your system resources. To see the status of the modules, expand **IBM Process Server v8.5.7** in the **Servers** view.



If you see any of the applications not in **Started** state, then in the **Servers** view right-click **IBM Process Server v8.5.7 at localhost** and select **Publish**.

You can also look for `Application started: [ApplicationName]` messages in the **Server Logs** view.

- __ 3. Open the `CreateApplication` user interface to start an instance of the `AccountVerification` business process.
 - __ a. Start Firefox as your browser.
 - __ b. Type the following address in the location bar (note the case) and press **Enter**:
`https://localhost:9443/AccountOpeningUI/Index.jsp`
 This address opens the login page for the user interface by using the IBM Process Server internal web container. No external web server is installed in your lab environment.
 - __ c. If the **This Connection is Untrusted** window is displayed, then expand **I Understand the Risks** and click **Add Exception**. Then, in the **Add Security Exception** window, click **Confirm Security Exception**.
 - __ d. At the **Login** prompt, type `admin` in the **Name** field and `web1sphere` in the **Password** field.
 - __ e. Click **Login**. This page (`Workplace.jsp`) is a JSP in the web project that you generated. All pages in the project are customizable.

- __ f. Under **Business Case**, click **New** to access the `CreateApplication` invocation task.



- __ g. `CreateApplication` is the only available task. Click the link.

The screenshot shows the 'Business Cases > New' page. It has a heading 'Business Cases > New' and a sub-instruction: 'Select a process or task for which you want to create a business case.' Below this, there's a dropdown menu labeled 'Task' with an option 'CreateApplication' which is highlighted with a red box.

- __ h. On the **CreateApplication** page, in the **Input Data** section, type `AbcCo` in the `companyName` field. You can leave the remaining fields blank. The `Determine Application Eligibility` Java snippet populates the remaining fields. The `companyName` determines the other values.

The screenshot shows the 'Business Cases > New > CreateApplication' page. It has a heading 'Business Cases > New > CreateApplication' and a sub-instruction: 'Enter the values for the input data and optionally provide additional information to create your task.' Below this, there's a section labeled 'Input Data' with several fields: 'accountNumber' (text box), 'applicationDate' (text box), 'applicationDecision' (checkbox), 'comments' (text box), and 'companyName' (text box containing 'AbcCo'). The 'companyName' field is highlighted with a red box.



Attention

The only available test cases are for the following company names: **AbcCo**, **IBM**, **ACME**, and **TestCo**. Do not deviate from these test cases, or you can receive unpredictable results.

- ___ i. Click **Create** at the bottom of the page. The web browser returns to the **Business Cases** page.
- ___ j. Minimize the browser window.
- ___ 4. In this test case, `eligibleApplication` is set to `false`. Therefore, the `Record Ineligible Application` activity is called. Verify that the application is written to the file system by the WebSphere Adapter for Flat Files.
 - ___ a. Open Windows Explorer.
 - ___ b. Browse to the directory where the output file was created (`C:\IneligibleAppArchive\outdir`).
 - ___ c. Two new files for the **AbcCo** test data are listed. One file maintains the sequence of the file, and the other contains the archive data. Open the `IneligibleApplication.n.txt` file, where `n` is the current value in the sequence.
 - ___ d. Examine the contents of the file. It contains the company name, comments, and other fields.
 - ___ e. Close the file.
 - ___ f. Close Windows Explorer.
- ___ 5. Verify the path that the application took through the business process by examining the messages in the **Server Logs** view.
 - ___ a. In IBM Integration Designer, switch to the **Server Logs** view.
 - ___ b. The following messages in the log confirm that the flow was successful:


```
Determine Applicant Eligibility - begins
Determine Applicant Eligibility - ends
Record Ineligible Application - begins
Account verification failed for customer: AbcCo
>>> Invoking Flat File Outbound service....
<<< Flat File Outbound service invoked OK!...
Record Ineligible Application - ends
```

Contents
[Java] Determine Applicant Eligibility - ends
[Java] Record Ineligible Application - begins
Account verification failed for customer : AbcCo
>>> Invoking the Flat File Outbound service ...
<<< Flat File Outbound service invoked OK! ...
[Java] Record Ineligible Application - ends

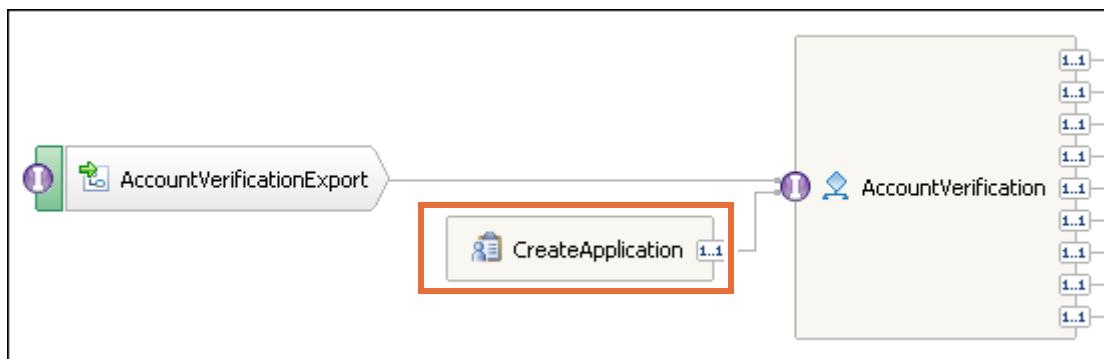
Part 2: Path two: Eligible applications with LOW credit risk

Overview

Eligible applications for customers that are deemed to be a low credit risk are automatically approved. When you test the solution by using company name **IBM**, the `eligibleApplication` attribute is set to `true`. When the Credit Check Service is called, a `creditScore` of `11` is returned, which represents a `LOW` `creditRisk`. When you submit an application with `companyName` **IBM**, the application flows through these activities: **Account Verification Receive > Determine Application Eligibility > Map to Credit Check > Credit Check Service > Map Credit Checking Result > Credit Risk Assessment > Create Output > Account Verification Reply**. Since the application is eligible and the `creditRisk` is `LOW`, the application is automatically approved. The `AccountVerification` process returns a reply.

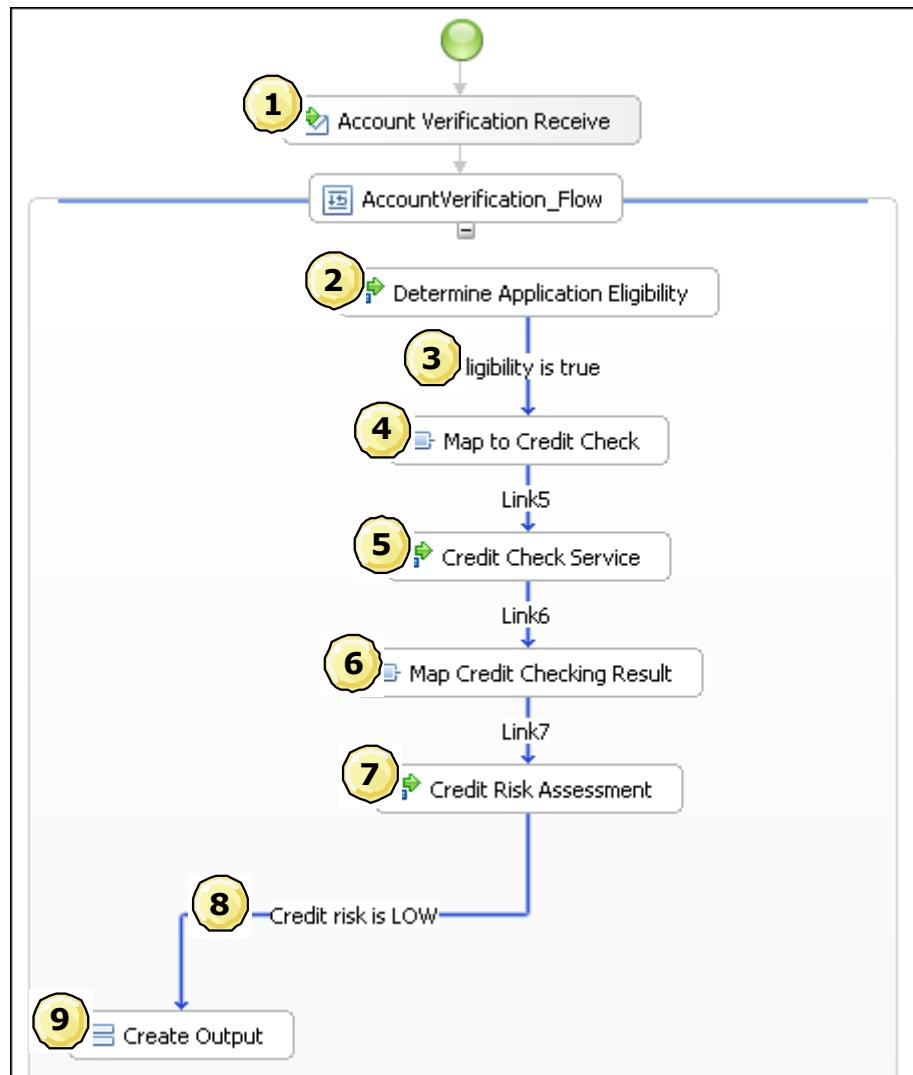
The second path through the process consists of the following steps:

- 1. The `AccountOpeningUI`, a JSF user interface for the `CreateApplication` human task, is used to create an application and to trigger a new instance of the `AccountVerification` business process.



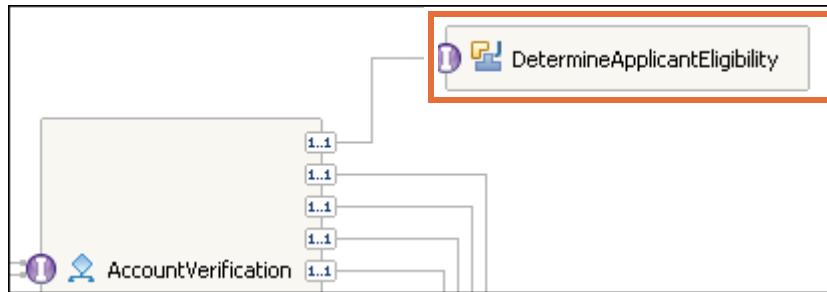
(Before the `CreateApplication` task is created, you use the IBM Integration Designer integrated test client to test your solution by calling the `AccountVerification` SCA component or the `AccountVerificationExport` component.)

- 2. When the application for **IBM** is received, it takes an eligible path through the process. The eligible path for this test case (a low credit risk) consists of the following activities:



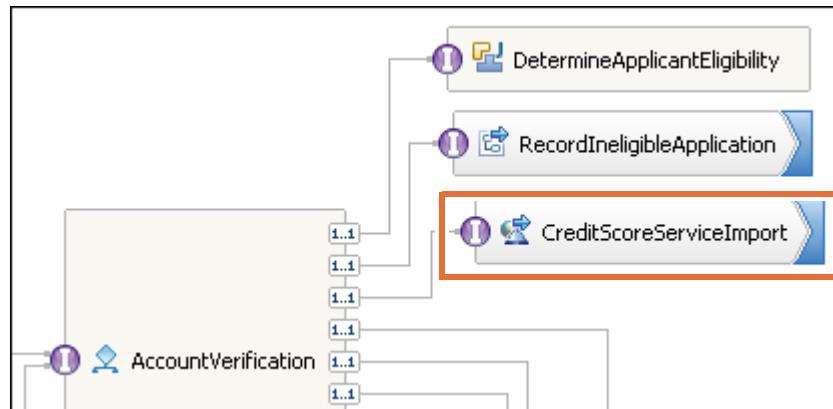
1. The `CreateApplication` invocation task sends the message (the application) to the `Account Verification Receive` activity.
The process follows the link to the next activity, the `AccountVerification_Flow` structured activity (a generalized flow).
2. The first activity in `AccountVerification_Flow` is the `Determine Application Eligibility` invoke activity. `Determine Application Eligibility` calls the `Determine Applicant Eligibility` service through the `Determine Application Eligibility` reference partner.

The Determine Applicant Eligibility service (in FoundationModule) is an SCA component, with a Java implementation that uses simple Java code to return sample data for each of the four test cases.

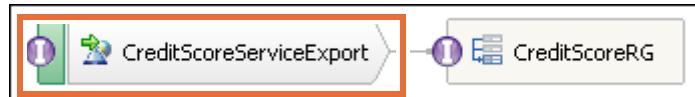


By default, the Java code sets the `eligibleApplication` element in `CustomerApplicationVariable` to false for **AbcCo** and to true for **ACME**, **IBM**, and **TestCo**. In practice, this element would not be set automatically.

3. The `eligibleApplication` element in `CustomerApplicationVariable` is examined. If `eligibleApplication` is true (which is always the case for **IBM**), then the “Eligibility is true” link is followed to the Map to Credit Check activity.
4. The Map to Credit Check activity is a data map activity that transforms the `CustomerApplication` business object into a `CreditCheckRequest` business object input. `CreditCheckRequest` is the input type for the `CreditScoreService`.
5. After the data is transformed, the link to the Credit Check Service invoke activity is followed. The Credit Check Service activity uses the `CreditCheckServicePartner` reference partner to call the `CreditScoreServiceImport` component.



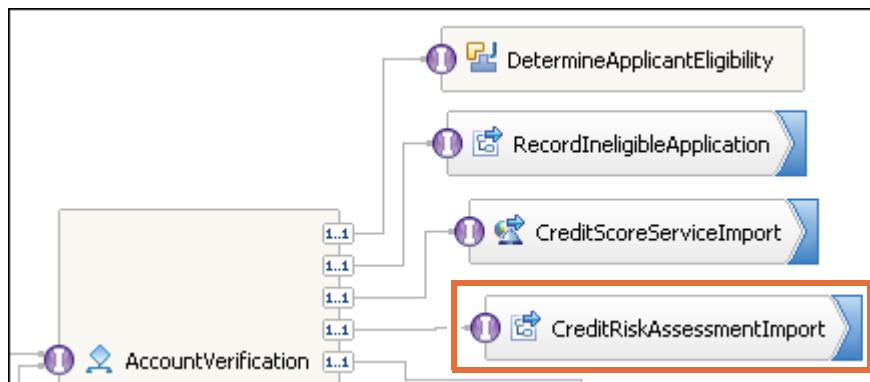
The `CreditScoreServiceImport` component calls the `CreditScoreServiceExport` component on the assembly diagram of the `CreditScoreService` module. The `CreditScoreServiceExport` is used to expose the services of the `CreditScoreRG` rule group.



The CreditScoreRG rule group contains a decision table that returns a fixed creditScore based on the companyName element in the CreditCheckRequest input business object. For IBM, the creditScore is always 11 (which represents a low creditRisk).

Conditions	"IBM"	"AbcCo"	"ACME"	Otherwise
request.companyName	11	1	6	1
calculateCreditScoreReturn.creditScore				

6. After the CreditScoreService returns the creditScore, the link to the Map Credit Checking Result data map activity is followed. The CustomerApplication variable is updated with the creditScore by transforming the data from a CreditCheckRequest business object to a CustomerApplication business object.
7. After the Map Credit Checking Result data map activity updates the CustomerApplication business object, the link to the Credit Risk Assessment invoke activity is followed. Credit Risk Assessment uses the CreditRiskAssessmentPartner reference partner to call the CreditRiskAssessmentImport component on the FoundationModule assembly diagram.



The CreditRiskAssessmentImport component calls the CreditRiskAssessmentExport component on the assembly diagram of the FoundationServices module. The CreditRiskAssessmentExport component exposes the services of the CreditRiskAssessment rule group.



The CreditRiskAssessment rule group contains a rule set that returns a creditRisk value that is based on the value in the creditScore element of the CustomerApplication. Because the creditScore for IBM is always 11, the rule set returns a value of Low for the

creditRisk field. The rule set updates the contents of the creditRisk element in the CustomerApplication business object with the value that the fired rule returned.

Name	RiskHIGH
Template	CreditRiskTemplate
Presentation	If the customer credit score is greater than 0 and less than 4 then the credit risk is HIGH
Name	RiskMED
Template	CreditRiskTemplate
Presentation	If the customer credit score is greater than 3 and less than 8 then the credit risk is MED
Name	RiskLOW
Template	CreditRiskTemplate
Presentation	If the customer credit score is greater than 7 and less than 12 then the credit risk is LOW

8. After the creditRisk element in the CustomerApplication variable is updated, the creditRisk is evaluated and since the risk is LOW, the “Credit risk is LOW” link is followed.
9. The final activity in this path is the CreateOutput assign activity. This activity assigns the value for the message element of the Message business object that the process returned. Because the creditRisk is LOW, the output message is: “Risk was LOW. Application automatically approved.”
10. When the message is assigned, the AccountVerification_flow is complete and the process follows the link to Account Verification Reply.

Account Verification Reply returns a Message business object to the client, which contains the message element that the Create Output activity assigned:

“Risk was LOW. Application automatically approved.”

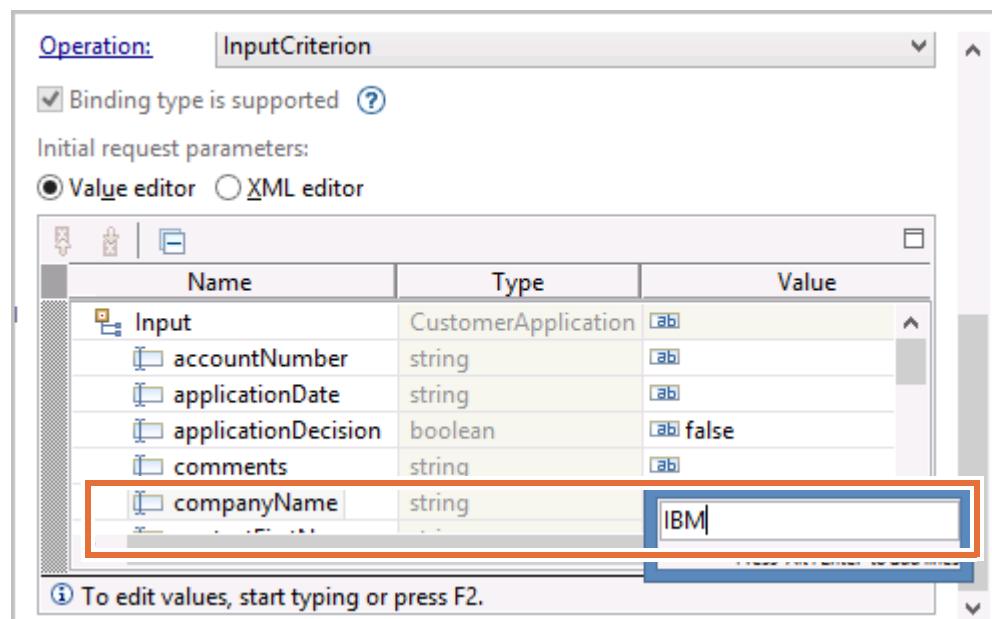


End-to-end test

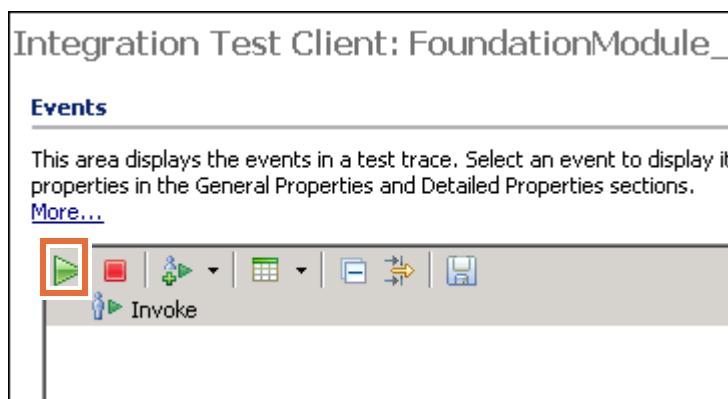
To test an eligible application with a LOW credit risk:

- 1. In IBM Integration Designer, in the Business Integration view, expand FoundationModule and double-click **Assembly Diagram**.
- The automatic approval process does not call any human tasks or write messages to the server log. You test the process in the integrated test environment so that you can use monitors to examine the test data.
- 2. Test the AccountVerificationExport component by using companyName **IBM**.
 - a. On the FoundationModule assembly diagram, right-click AccountVerificationExport and choose **Test Component** from the menu.

- ___ b. When the integration test client opens, in the **Initial request parameters** section, type IBM in the `companyName` field.

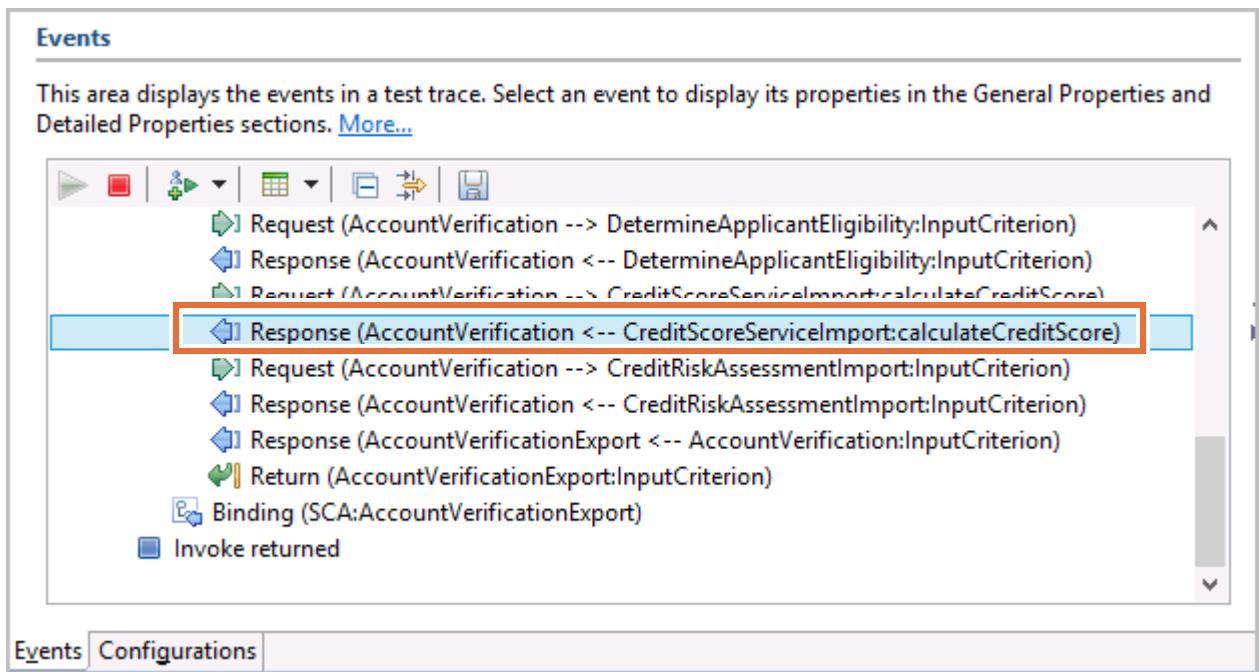


- ___ c. Click the **Continue** icon on the **Events** window toolbar.



- ___ d. If a **Deployment Location** window opens, select **IBM Process Server v8.5.7 at localhost** and click **Finish**.
- ___ e. At the **User Login** window, accept the default entries for **User ID** and **Password**, and click **OK**. These fields are set to `admin` and `web1sphere` by default.
- ___ f. The test run is complete when you receive the blue, square stop node in the **Events** window.

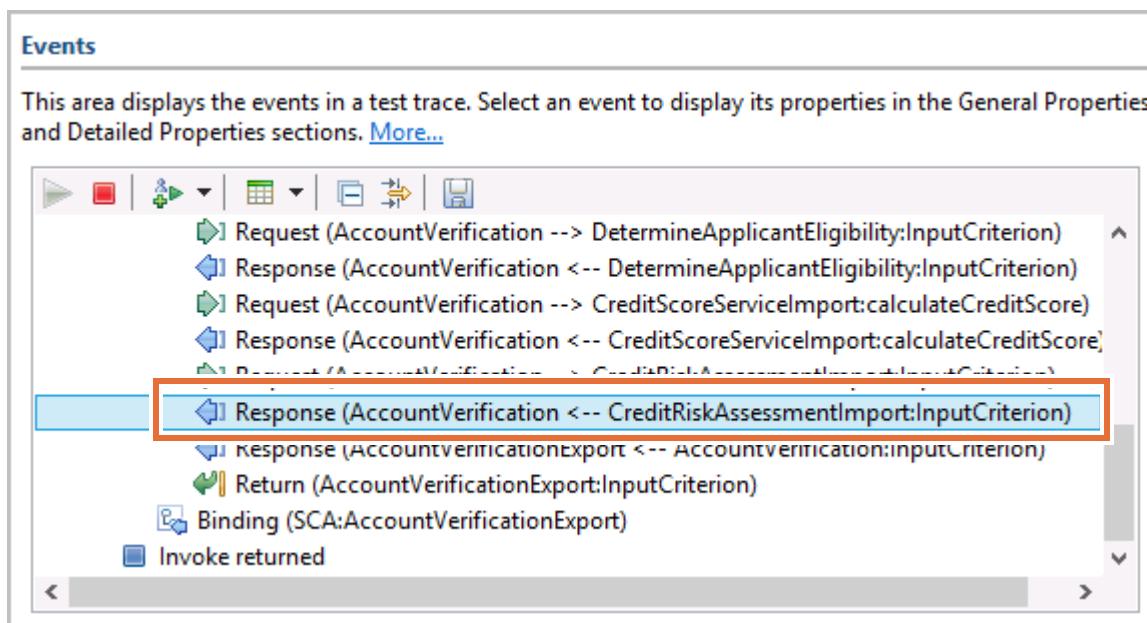
- __ g. In the **Events** window, click the **Response (AccountVerification <-- CreditScoreServiceImport:calculateCreditScore)** event.



- __ h. In the **Response parameters** section, the predetermined creditScore for IBM is 11.

Name	Type	Value
calculateCreditScoreReturn	CreditCheckRequest	
accountNumber	string	IBM007
companyName	string	IBM
creditScore	int	11
dateRequested	string	2016-12-22-05:00

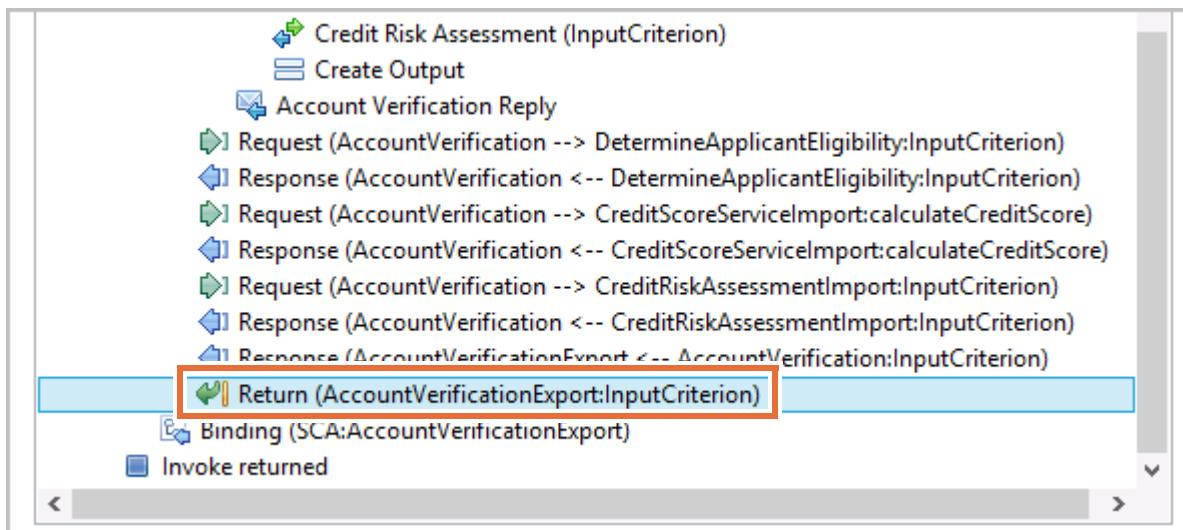
- i. In the **Events** window, click the **Response (AccountVerification <-- CreditRiskAssessmentImport:InputCriterion)** event.



- j. Because the `creditScore` is 11, the `creditRisk` in the **Response parameters** section is `LOW`.

Name	Type	Value
Output	CustomerApplication	[ab]
accountNumber	string	[ab] IBM007
applicationDate	string	[ab] Dec 22, 2016
applicationDecision	boolean	[ab] true
comments	string	[ab] None
companyName	string	[ab] IBM
contactFirstName	string	[ab] Landon
contactLastName	string	[ab] Donovan
contactPhoneNumber	string	[ab] 547-555-3172
creditRating	string	[ab] A++
creditReportNeeded	boolean	[ab] true
creditRisk	string	[ab] LOW <i>(highlighted with a red box)</i>
creditScore	int	[ab] 11
customerCity	string	[ab] Boston
customerCountry	string	[ab] USA
eligibleApplication	boolean	[ab] true
ineligibleReason	string	[ab]
pricingCode	string	[ab] 34
pricingScore	string	[ab] 32

- k. Since the `creditRisk` is `LOW`, the application is automatically approved. Click the **Return (AccountVerificationExport:InputCriterion)** event.



- l. In the **Return parameters** section, the output message is: Risk was `LOW`. Application automatically approved. You can adjust the **Value** column width to view the entire message.

Return parameters		
	Name	Type
Output	message	string
		Risk was <code>LOW</code> . Application automatically approved...

- m. Close the `FoundationModule_Test` tab and click **No** when you are prompted to save the test trace.

Part 3: Path three: Eligible applications with **HIGH** credit risk

Overview

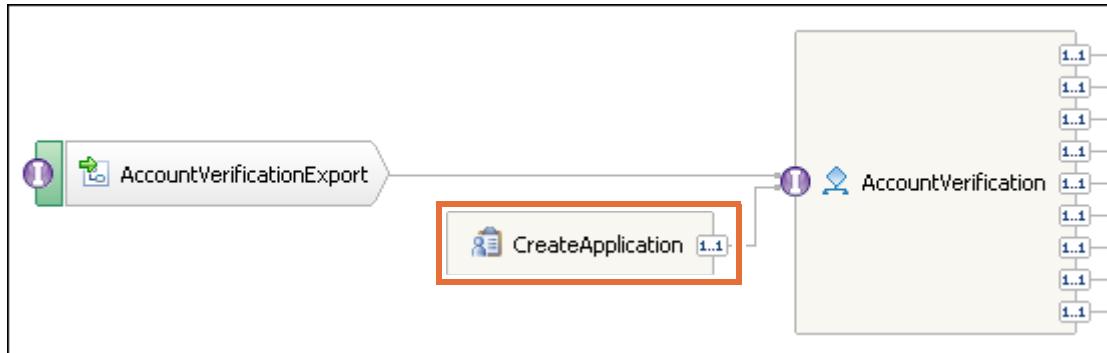
Eligible applications for customers that are deemed to be a high credit risk require supplemental documentation and manual approval or denial. When you test the solution by using company name **TestCo**, the `eligibleApplication` attribute is set to `true` and the `creditRisk` evaluates to `HIGH` (the `creditScore` returned is `1`). When you submit an application by using `companyName TestCo`, the application flows through these activities: **Account Verification Receive** > **Determine Application Eligibility** > **Map to Credit Check** > **Credit Check Service** > **Map Credit Checking Result** > **Credit Risk Assessment** > **Assign Variable** > **While More Documents Required** > **Request More Documentation**.

A user interface for **Request More Documentation** is used to change the `comment` field from `None` to `Complete`. After the **While More Documents Required** loop is left, the application flows

through **Merge Assign > Final Application Review**. A user interface for `Final Application Review` is used to update the `applicationDecision` field. If `applicationDecision` is `true`, the application flows through **Create Output > Account Verification Reply**. If `applicationDecision` is `false`, the application flows through **Generate Decline > Record Declined Application > Create Output > Account Verification Reply**.

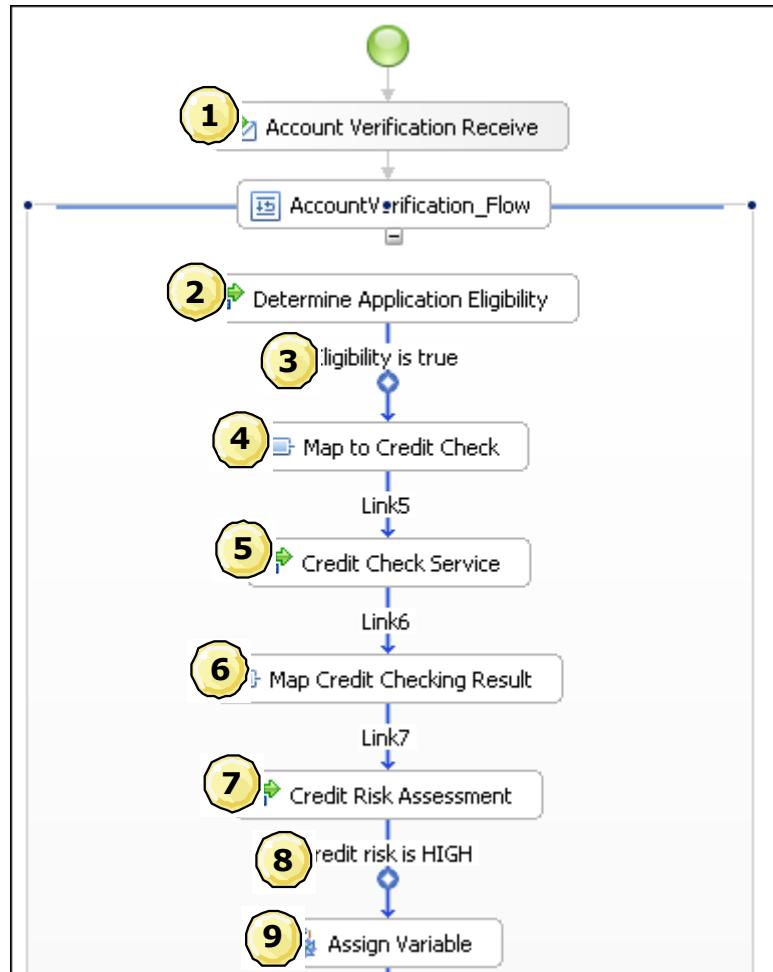
The third path through the process consists of the following steps:

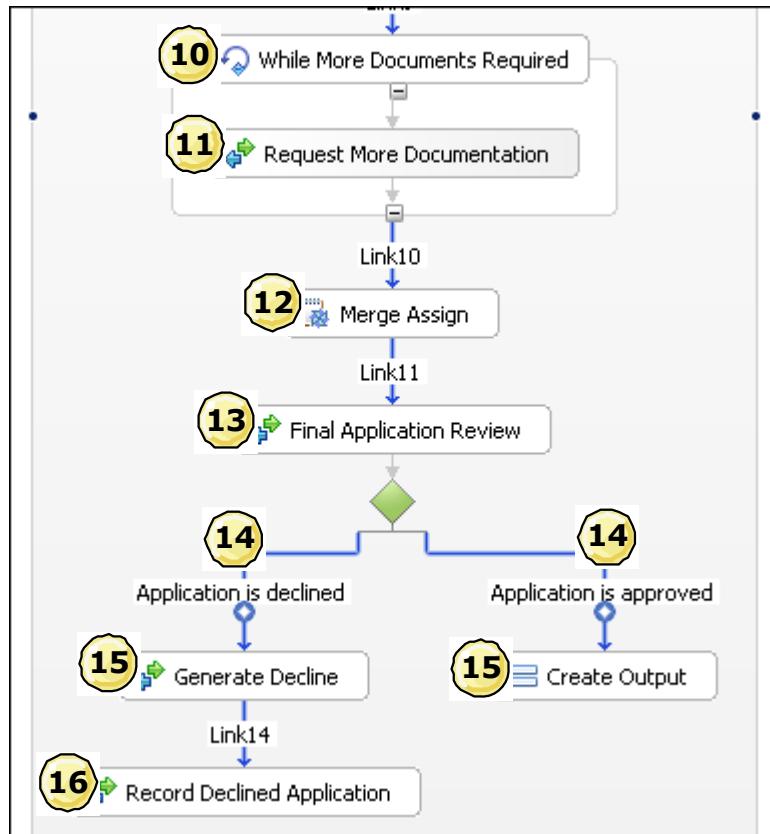
- 1. The `AccountOpeningUI`, a JSF user interface for the `CreateApplication` human task, is used to create an application and to trigger a new instance of the `AccountVerification` business process.



(Before the `CreateApplication` task is created, you use the IBM Integration Designer integrated test client to test your solution by calling the `AccountVerification` SCA component or the `AccountVerificationExport` component.)

- 2. When the application for **TestCo** is received, it takes an eligible path through the process. The eligible path for this test case (a high credit risk) consists of the following activities (the diagram is split for readability):

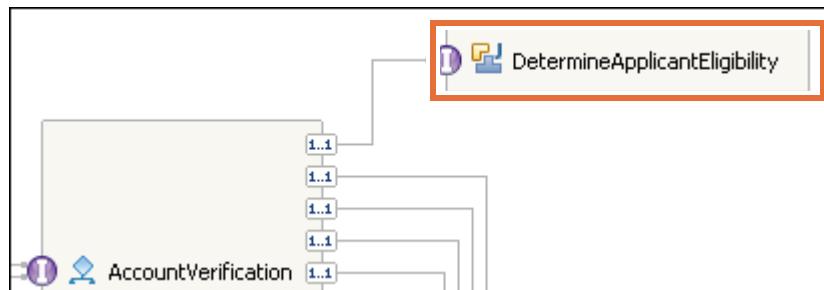




1. The Account Verification Receive activity receives the message (the application) that the CreateApplication invocation task sent. The process follows the link to the next activity, the AccountVerification_Flow structured activity (a generalized flow).

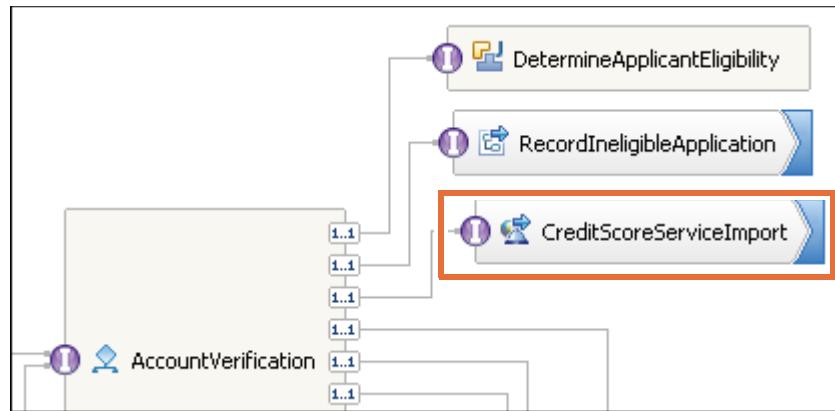
The first activity in AccountVerification_Flow is the Determine Application Eligibility invoke activity. Determine Application Eligibility calls the Determine Applicant Eligibility service through the Determine Application Eligibility reference partner.

2. The Determine Applicant Eligibility service (in FoundationModule) is an SCA component, with a Java implementation that uses simple Java code to return sample data for each of the four test cases.



By default, the Java code sets the `eligibleApplication` element in the `CustomerApplicationVariable` to `false` for **AbcCo** and to `true` for **ACME**, **IBM**, and **TestCo**. In practice, this element would not be set automatically; a human might likely set it.

3. The `eligibleApplication` element in `CustomerApplicationVariable` is examined. If `eligibleApplication` is `true` (which is always the case for **TestCo**), then the “Eligibility is true” link is followed to the `Map to Credit Check` activity.
4. The `Map to Credit Check` activity is a data map activity that transforms the `CustomerApplication` business object into a `CreditCheckRequest` business object input. `CreditCheckRequest` is the input type for the `CreditScoreService`.
5. After the data is transformed, the link to the `Credit Check Service` invoke activity is followed. The `Credit Check Service` activity uses the `CreditCheckServicePartner` reference partner to call the `CreditScoreServiceImport` component.



The `CreditScoreServiceImport` component calls the `CreditScoreServiceExport` component on the assembly diagram of the `CreditScoreService` module. The `CreditScoreServiceExport` is used to expose the services of the `CreditScoreRG` rule group.



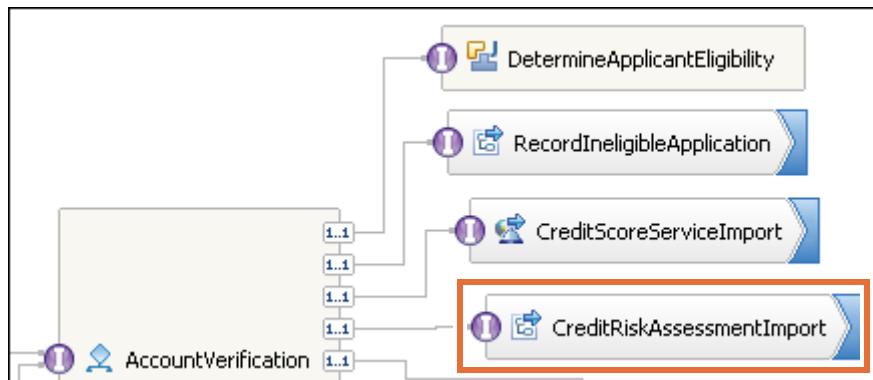
The `CreditScoreRG` rule group contains a decision table that returns a fixed `creditScore` based on the `companyName` element in the `CreditCheckRequest` input business object. For **TestCo**, the `creditScore` is always 1 because the “otherwise” condition is evaluated (which represents a high `creditRisk`).

Conditions					Actions
request.companyName	"IBM"	"AbcCo"	"ACME"	Otherwise	
calculateCreditScoreReturn.creditScore	11	1	6	1	

After the `CreditScoreService` returns the `creditScore` value, the link to the `Map Credit Checking Result` data map activity is followed. The `CustomerApplication` variable is updated with the `creditScore` by transforming the data from a `CreditCheckRequest` business object to a `CustomerApplication` business object.

6. After the `Map Credit Checking Result` data map activity updates the `CustomerApplication` business object, the link to the `Credit Risk Assessment` invoke activity is followed. Credit

Risk Assessment uses the CreditRiskAssessmentPartner reference partner to call the CreditRiskAssessmentImport component on the FoundationModule assembly diagram.



The CreditRiskAssessmentImport component calls the CreditRiskAssessmentExport component on the assembly diagram of the FoundationServices module. The CreditRiskAssessmentExport component exposes the services of the CreditRiskAssessment rule group.



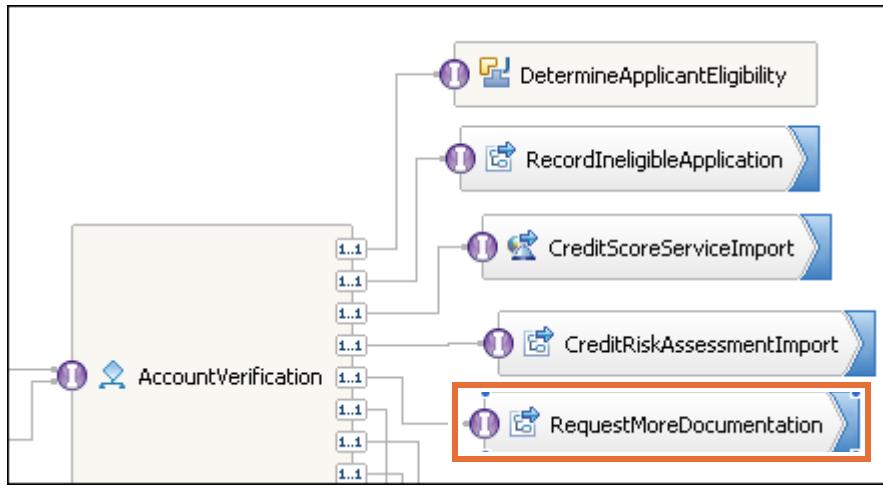
7. The CreditRiskAssessment rule group contains a rule set that returns a creditRisk value that is based on the value in the creditScore element of the CustomerApplication. Because the creditScore for **TestCo** is always 1, the rule set returns a value of HIGH for the creditRisk field. The rule set updates the contents of the creditRisk element in the CustomerApplication business object with the value that the fired rule returned.

Name	RiskHIGH
Template	CreditRiskTemplate
Presentation	If the customer credit score is greater than 0 and less than 4 then the credit risk is HIGH
Name	RiskMED
Template	CreditRiskTemplate
Presentation	If the customer credit score is greater than 3 and less than 8 then the credit risk is MED
Name	RiskLOW
Template	CreditRiskTemplate
Presentation	If the customer credit score is greater than 7 and less than 12 then the credit risk is LOW

8. After the creditRisk element in the CustomerApplication variable is updated, the creditRisk is evaluated. Because the risk is HIGH, the “Credit risk is HIGH” link is followed.
9. After the “Credit risk is HIGH” link is followed, the Assign Variable activity is processed. Assign Variable is a Java snippet that generates a new CustomerApplication business object (CustomerApplicationVariable2) and assigns the contents of the original CustomerApplication variable to it.

10. After the new variable is created and populated, the link to `While More Documents Required` is followed. `While More Documents Required` is a loop that processes the activities in its scope while the `comments` element of `CustomerApplicationVariable2` is set to `None`.

The only activity inside the `While More Documents Required` while loop is the `Request More Documentation` invoke activity. `Request More Documentation` uses the `RequestMoreDocumentationPartner` reference partner to call the `RequestMoreDocumentation` import component on the `FoundationModule` assembly diagram.



11. The `RequestMoreDocumentation` import component starts the `RequestMoreDocumentationExport` component on the assembly diagram of the `HumanTaskServices` module.

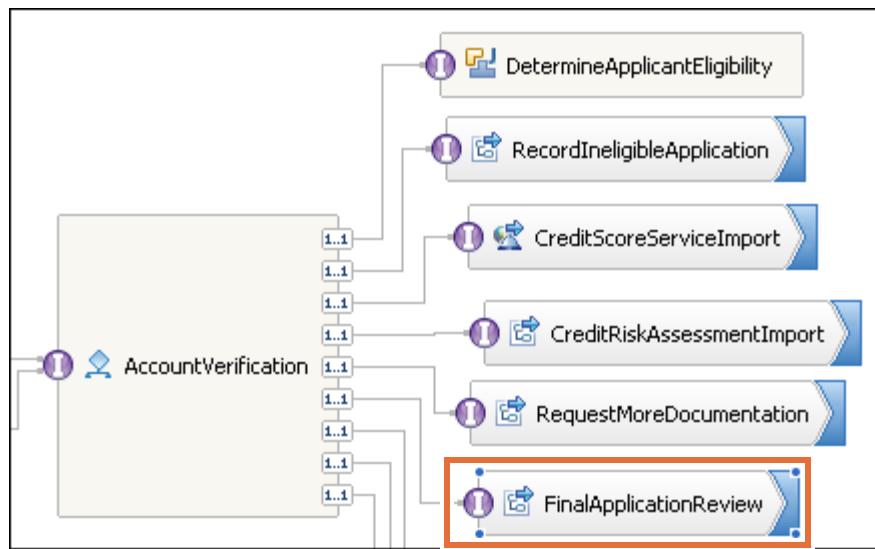


The `RequestMoreDocumentationExport` component exposes the services of the `RequestMoreDocumentation` stand-alone human task. Using a human task user interface such as the Business Process Choreographer Explorer, an employee updates the `comments` field from `None` to `Complete`, indicating that the supplemental documentation that the high-risk application requires was received.

After the `comments` field is updated, the loop is broken and the link to `Merge Assign` is followed. `Merge Assign` is a Java snippet that creates a variable (`CustomerApplicationVariable`) and assigns the contents of `CustomerApplicationVariable2` to it.

12. After `Merge Assign` is complete, the link to `Final Application Review` is followed. `Final Application Review` is an invoke activity that uses the `FinalApplicationReviewPartner`

reference partner to call the `FinalApplicationReview` import component on the `FoundationModule` assembly diagram.



13. The `FinalApplicationReview` import starts the `FinalApplicationReviewExport` component on the `HumanTaskServices` assembly diagram.

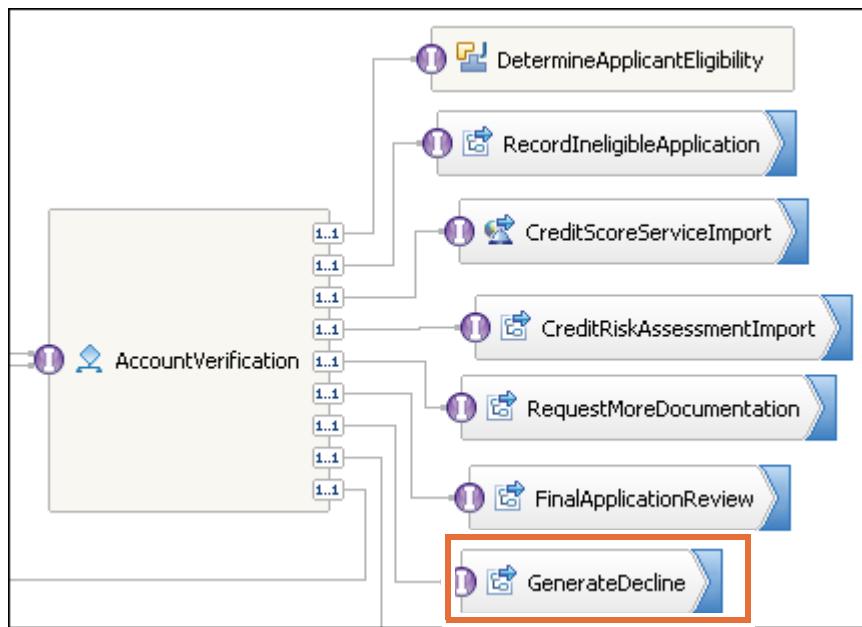


The `FinalApplicationReviewExport` component exposes the services of the `FinalApplicationReview` stand-alone human task. Using a human task user interface such as `AccountProcessingUI`, an employee does a final review of the high-risk application and supplemental documentation to decide whether the application must be approved or declined. Using the user interface, the employee updates the `applicationDecision` field to `true` (approved) or `false` (declined).

Depending on the action that the employee takes, the `applicationDecision` field is examined, and either the “Application is declined” link is followed or the “Application is approved” link is followed.

14. If the “Application is approved” link is followed, the `Create Output` activity is processed. `Create Output` is an assign activity that assigns the `message` element of the `Message` business object to: “Application was approved.”
15. If the “Application is declined” link is followed, the `Generate Decline` invoke activity is processed. The `Generate Decline` activity uses the `GenerateDeclinePartner` reference

partner to call the `GenerateDecline` import component on the `FoundationModule` assembly diagram.



The `GenerateDecline` import component calls the `RouteRequestExport` component on the `RouterMediationService` assembly diagram.



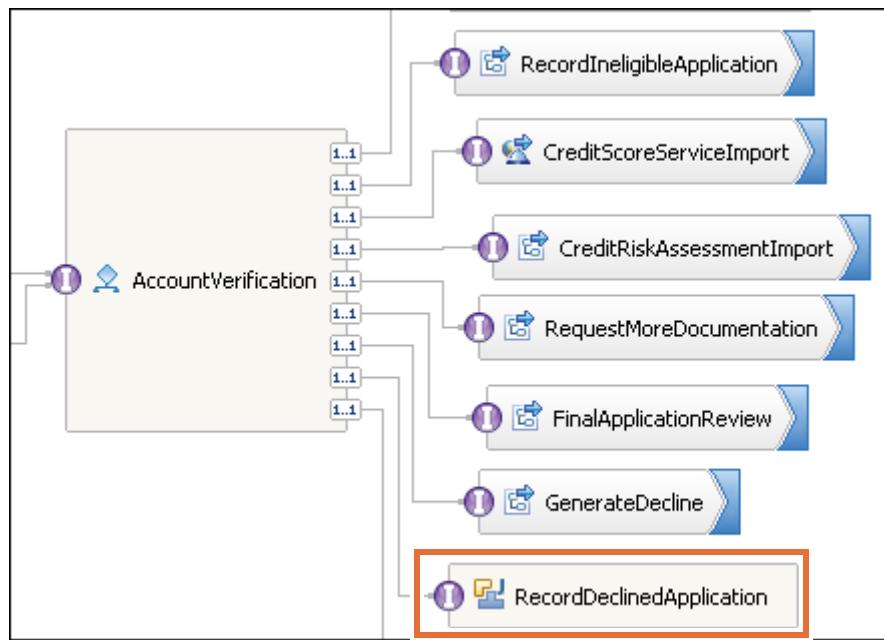
`RouteRequestExport` exposes the services of the `RouteRequest` mediation flow. If the `creditRisk` is `HIGH` (which is the case for `TestCo`) and `applicationDecision` is `false`, the message is routed to the `GenerateDeclineService` import component. The `GenerateDeclineService` import calls the `GenerateDeclineExport` component on the `FoundationServices` assembly diagram.



`GenerateDeclineExport` exposes the services of the `GenerateDecline` Java component. This component sets the `message` element of the `Message` business object to: "Account for customer TestCo was declined and the credit risk was HIGH."

16. After the GenerateDecline activity is processed, the link to Record Declined Application is followed. The Record Declined Application invoke activity uses the

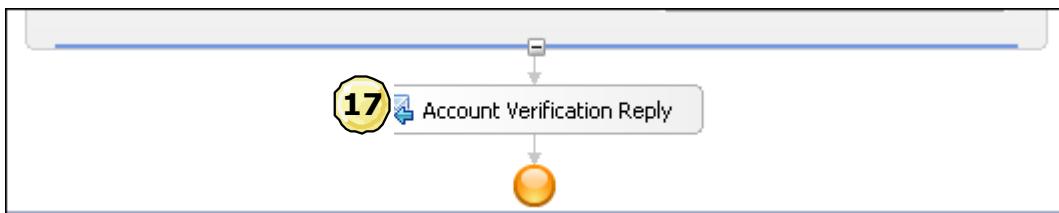
`RecordDeclinedApplicationPartner` reference partner to call the `RecordDeclinedApplication` Java component on the `FoundationModule` assembly diagram.



`RecordDeclinedApplication` writes messages to the console and to the server log, indicating that the application was declined.

When the message is assigned through the `Create Output` activity, or the `RecordDeclinedApplication` is processed, `AccountVerification_flow` is complete, and the process follows the link to `Account Verification Reply`.

17. `Account Verification Reply` returns one of two messages to the client:
If the application is approved, a `Message` business object is returned to the client, which contains the `message` element that the `Create Output` assigned: "Application was approved."
If the application is denied, a `Message` business object is returned to the client, which contains the `message` element that the `GenerateDecline` service assigned: "Account for customer TestCo was declined and the credit risk was HIGH."



End-to-end test

To test an eligible application with a HIGH credit risk:

- 1. Create a business case in the business user client.

In the ineligible application test, you opened a Firefox browser window to the business user client.

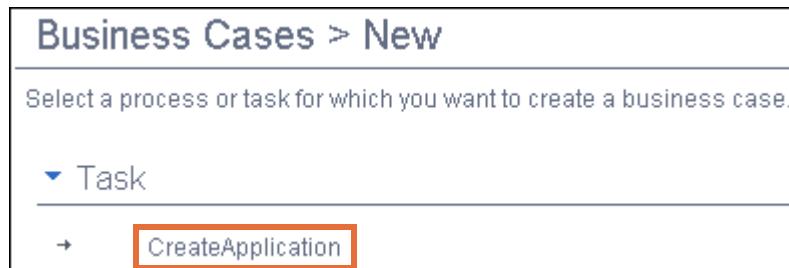
<https://localhost:9443/AccountOpeningUI/Index.jsp>

If this session is expired (to check whether it is expired, click **Home** and refresh your browser), you must log in again to create another case. Use `admin` in the **User** field and `web1sphere` in the **Password** field to log in.

- 2. Enter data for the test case in the `CreateApplication` user interface.
 - a. Under **Business Case**, click **New** to access the `CreateApplication` invocation task.



- b. `CreateApplication` is the only available task. Click the link.



- ___ c. On the **CreateApplication** page, in the **Input Data** section, type `TestCo` in the **companyName** field. You can leave the remaining fields blank. The **Determine Application Eligibility** Java snippet populates the remaining fields that are based on `companyName`.

Business Cases > New > CreateApplication

Enter the values for the input data and optionally provide additional information to create your task.

▼ Input Data

accountNumber	<input type="text"/>
applicationDate	<input type="text"/>
applicationDecision	<input type="checkbox"/>
comments	<input type="text"/>
companyName	<input type="text" value="TestCo"/>



Note

The only available test cases are for the following company names: **AbcCo**, **IBM**, **ACME**, and **TestCo**. Do not deviate from these test cases, or you can receive unpredictable results.

- ___ d. Click **Create** at the bottom of the page. The browser window returns to the **Business Cases** page.
- ___ e. Minimize the browser.
- ___ 3. After you submit an application by using **companyName** `TestCo`, the application flows through the process to **Request More Documentation**. Using the Business Process Choreographer Explorer client, change the **comment** field from **None** to **Complete** to terminate the **Request More Documentation** loop.
- ___ a. In IBM Integration Designer, switch to the **Servers** view.
- ___ b. Right-click **IBM Process Server v8.5.7** and choose **Launch > Business Process Choreographer Explorer** from the menu.
- ___ c. If you receive a security alerts window, click **Yes** twice to proceed.
- ___ d. At the login pane, type `admin` in the **User Name** field and `web1sphere` in the **Password** field.
- ___ e. Click **Login**. The **My To-dos** page opens.

- f. A **Request More Documentation** task is ready for you to claim. Click the **Request More Documentation** link.

The screenshot shows a user interface titled 'My To-dos'. At the top, there are several buttons: 'Work on', 'Release', 'Transfer', 'Start', 'Change Business Category', and 'Refresh'. Below these buttons is a search bar with dropdown menus for 'Priority', 'Task Name', 'State', 'Kind', 'Owner', and 'Originator'. The search bar also displays the number '5' and the text 'Request More Documentation'. The main area shows a single task card with the title 'Request More Documentation', status 'Ready', and owner 'admin'.

- g. On the **Task Instance** page, click **Work on** to claim the task.
- h. On the **Task Message** page, scroll to the **Task Output Message** section and click **Edit Source** (at the bottom of the section).
- i. In Windows Explorer, browse to the directory that contains the support files for this exercise (C:\labfiles\Support Files\EX1).
- j. Open EX1_Test_Data.txt in a text editor such as Notepad.
- k. Copy the text in EX1_Test_Data.txt and paste it over the existing text in the **Task Output Message Source View** window.

The screenshot shows a window titled 'Source View*'. It contains an XML document with various fields filled in. At the bottom of the window are four buttons: 'Cancel', 'Validate', 'Confirm', and 'Reset'.

```

< xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:p="http://FoundationLibrary/RequestMoreDocumentation">
<Output>
  <accountNumber>TEST001</accountNumber>
  <applicationDate>Oct 02, 2014</applicationDate>
  <applicationDecision>false</applicationDecision>
  <comments>Complete</comments>
  <companyName>TestCo</companyName>
  <contactFirstName>Jane</contactFirstName>
  <contactLastName>Doe</contactLastName>
  <contactPhoneNumber>872-555-9915</contactPhoneNumber>
  <creditRating>C</creditRating>
  <creditReportNeeded>true</creditReportNeeded>
  <creditRisk>HIGH</creditRisk>
  <creditScore>1</creditScore>
  <customerCity>Chicago</customerCity>
  <customerCountry>USA</customerCountry>
  <eligibleApplication>true</eligibleApplication>
  <ineligibleReason>Uncertain collateral</ineligibleReason>

```

- __ l. Click **Confirm**.
- __ m. When you are returned to the **Form View**, verify that the comments field in the **Task Output Message** area is set to: Complete

Form View*

Output	
accountNumber	TEST001
applicationDate	Oct 02, 2014
applicationDecision	<input type="checkbox"/> Remove
comments	Complete
companyName	TestCo
contactFirstName	Jane
contactLastName	Doe

- __ n. At the top of the page, click **Complete** to finish the task.
- __ o. Leave Windows Explorer open and close `EX1_Test_Data.txt`.
- __ 4. The Final Application Review task is used to update the applicationDecision field. If applicationDecision is true, the application flows through **Create Output > Account Verification Reply**. If applicationDecision is false, the application flows through **Generate Decline > Record Declined Application > Create Output > Account Verification Reply**. Using the Business Process Choreographer Explorer client, change applicationDecision to false.
 - __ a. After a moment, click the **My To-dos** link in the **Task Instances** section of the Business Process Choreographer Explorer client.



- __ b. On the **My To-dos** page, click the **check box** for the **Final Application Review** task, and click **Work On**.

My To-dos

Use this page to work on tasks that are assigned to you. [i](#)

Work on [Release](#) [Transfer](#) [Start](#) [Change Business Category](#) [Refresh](#)

<input type="checkbox"/>	Priority	Task Name	State	Kind	Owner	Originator
<input checked="" type="checkbox"/>	5	Final Application Review	Ready	To-do Task	admin	

- __ c. On the **Task Message** page, scroll to the **Task Output Message** section and click **Edit Source** (at the bottom of the section).
- __ d. In Windows Explorer, open `C:\labfiles\Support Files\EX1\EX1_Test_Data2.txt` in a text editor such as Notepad (note the file extension `.txt`).
- __ e. Copy the text in `EX1_Test_Data2.txt` and paste it over the existing text in the **Task Output Message Source View** window.

Source View

```

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:p="http://FoundationLibrary/FinalApplicationReview">
<Output>
  <accountNumber>TEST001</accountNumber>
  <applicationDate>Oct 02, 2014</applicationDate>
  <applicationDecision>false</applicationDecision>
  <comments>Complete</comments>
  <companyName>TestCo</companyName>
  <contactFirstName>Jane</contactFirstName>
  <contactLastName>Doe</contactLastName>
  <contactPhoneNumber>872-555-9915</contactPhoneNumber>
  <creditRating>C</creditRating>
  <creditReportNeeded>true</creditReportNeeded>
  <creditRisk>HIGH</creditRisk>
  <creditScore>1</creditScore>
  <customerCity>Chicago</customerCity>
  <customerCountry>USA</customerCountry>
  <eligibleApplication>true</eligibleApplication>
  <ineligibleReason>Uncertain collateral</ineligibleReason>

```

- __ f. Click **Confirm**.

- __ g. When you are returned to the **Form View**, verify that the `applicationDecision` field in the **Task Output Message** area is set to `false` (it is not checked).

Form View*		
Output		
	accountNumber	TEST001
	applicationDate	Oct 02, 2014
	applicationDecision	<input type="checkbox"/> Remove
	comments	Complete
	companyName	TestCo
	contactFirstName	Jane
	contactLastName	Doe

- __ h. At the top of the page, click **Complete**.
- __ 5. Verify the path that the application took through the remainder of the business process by examining the messages in the **Server Logs** view.
- __ a. In IBM Integration Designer, switch to the **Server Logs** view.
- __ b. The following messages in the server log confirm a successful path through the business process.

```
Generate Decline - begins
Generate Decline - Account for customer TestCo was declined and the credit
risk was HIGH
Generate Decline - ends
Record Declined Application - begins
Record Declined Application - ends
```

```
[Java] Determine Applicant Eligibility - begins
[Java] Determine Applicant Eligibility - ends
[Java] Generate Decline - begins
[Java] Generate Decline - Account for customer TestCo was declined and the credit risk was HIGH
[Java] Generate Decline - ends
[Java] Record Declined Application - begins
[Java] Record Declined Application - ends
```



Note

If you cannot read the log message contents, double-click the message.

**Optional**

If time allows, run the test again with `applicationDecision` set to `true`. For this test case, `Generate Decline` is not started, so the `Generate Decline` messages are not shown in the **Server Logs** view.

- 6. Click the **Logout** link to log out of the Business Process Choreographer Explorer. Do not close the tab. It is used to test the final path.

Part 4: Path four: Eligible applications with MED credit risk

Overview

Eligible applications for customers that are deemed to be a medium credit risk also require supplemental documentation and manual approval or denial. When you test the applications by using company name **ACME**, the `eligibleApplication` attribute is set to `true` and the `creditRisk` evaluates to **MED** (the `creditScore` returned is 6). When you submit an application by using `companyName ACME`, the application flows through these activities: **Account Verification Receive > Determine Application Eligibility > Map to Credit Check > Credit Check Service > Map Credit Checking Result > Credit Risk Assessment > Assign Variable > While More Documents Required > Request More Documentation**.

A user interface for `Request More Documentation` is used to change the `comment` field from `None` to `Complete`. After leaving the `While More Documents Required` loop, the application flows through **Merge Assign > Create Output > Final Application Review**. A user interface for `Final Application Review` is used to update the `applicationDecision` field. If `applicationDecision` is `true`, the application flows to **Create Output > Account Verification Reply**. If `applicationDecision` is `false`, the application flows through **Generate Decline > Record Declined Application > Create Output > Account Verification Reply**.

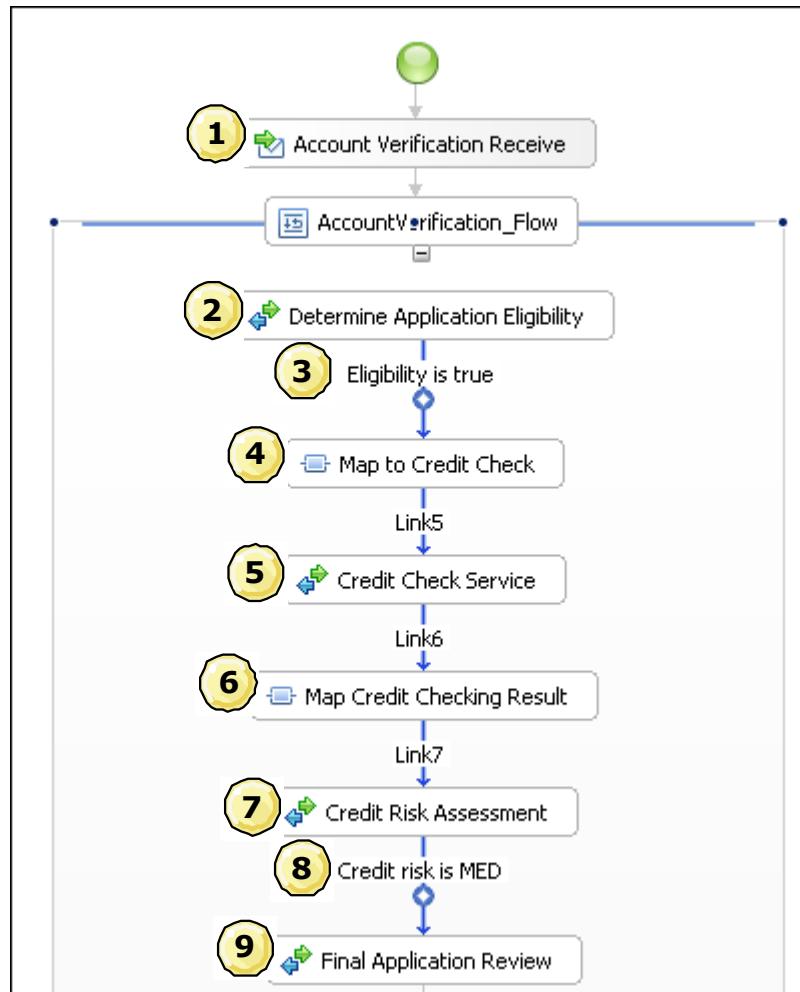
The fourth path through the process consists of the following steps:

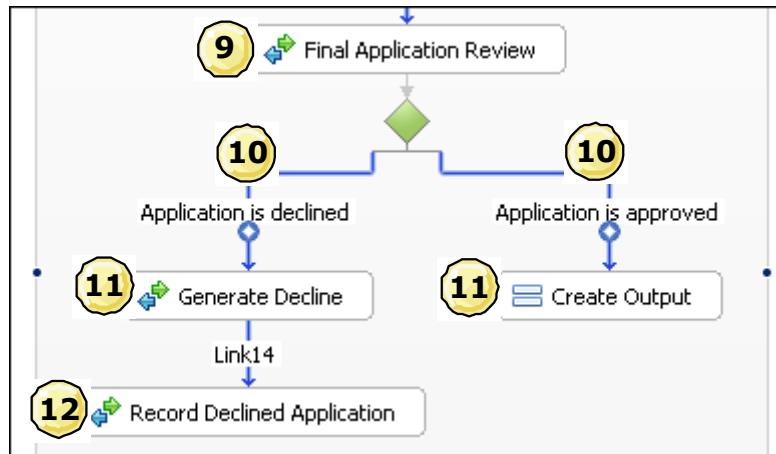
- 1. The `AccountOpeningUI`, a JSF user interface for the `CreateApplication` human task, is used to create an application and to trigger a new instance of the `AccountVerification` business process.



(Before the `CreateApplication` task is created, you use the IBM Integration Designer integrated test client to test your solution by calling the `AccountVerification SCA` component or the `AccountVerificationExport` component.)

- 2. When the application for **TestCo** is received, it takes an eligible path through the process. The eligible path for this test case (a high credit risk) consists of the following activities (the diagram is split for readability):



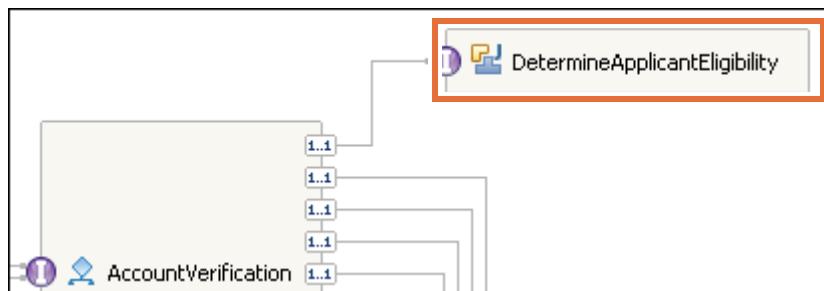


1. The Account Verification Receive activity receives the message (the application) that the CreateApplication invocation task sent.

The process follows the link to the next activity, the `AccountVerification_Flow` structured activity (a generalized flow).

2. The first activity in `AccountVerification_Flow` is the Determine Application Eligibility invoke activity. Determine Application Eligibility calls the Determine Applicant Eligibility service through the Determine Application Eligibility reference partner.

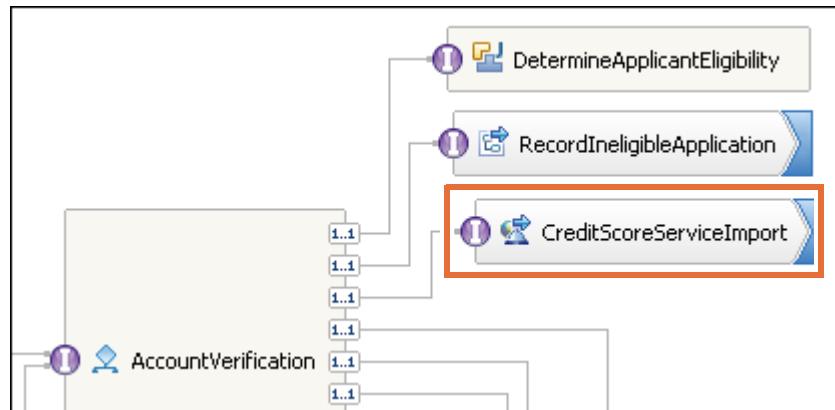
The Determine Applicant Eligibility service (in the `FoundationModule`) is an SCA component, with a Java implementation that uses simple Java code to return sample data for each of the four test cases.



By default, the Java code sets the `eligibleApplication` element in `CustomerApplicationVariable` to `false` for **AbcCo** and to `true` for **ACME**, **IBM**, and **TestCo**. In practice, this element would not be set automatically; a person would likely set it.

3. The `eligibleApplication` element in `CustomerApplicationVariable` is examined. If `eligibleApplication` is `true` (which is always the case for **ACME**), then the “Eligibility is true” link is followed to the Map to Credit Check activity.
4. The Map to Credit Check activity is a data map activity that transforms the `CustomerApplication` business object into a `CreditCheckRequest` business object input. `CreditCheckRequest` is the input type for the `CreditScoreService`.

- After the data is transformed, the link to the Credit Check Service invoke activity is followed. The Credit Check Service activity uses the CreditCheckServicePartner reference partner to call the CreditScoreServiceImport component.



The `CreditScoreServiceImport` component calls the `CreditScoreServiceExport` component on the assembly diagram of the `CreditScoreService` module. The `CreditScoreServiceExport` is used to expose the services of the `CreditScoreRG` rule group.

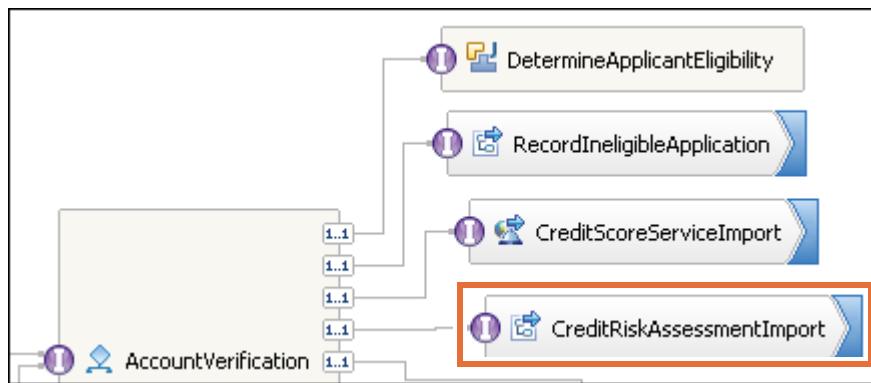


The `CreditScoreRG` rule group contains a decision table that returns a fixed `creditScore` based on the `companyName` element in the `CreditCheckRequest` input business object. For **ACME**, the `creditScore` is always 6 (which represents a medium `creditRisk`).

Conditions				
request.companyName	"IBM"	"AbcCo"	"ACME"	Otherwise
calculateCreditScoreReturn.creditScore	11	1	6	1

6. After the CreditScoreService returns the creditScore value, the link to the Map Credit Checking Result data map activity is followed. The CustomerApplication variable is updated with the creditScore by transforming the data from a CreditCheckRequest business object to a CustomerApplication business object.
 7. After the Map Credit Checking Result data map activity updates the CustomerApplication business object, the link to the Credit Risk Assessment invoke activity is followed. Credit

Risk Assessment uses the CreditRiskAssessmentPartner reference partner to call the CreditRiskAssessmentImport component on the FoundationModule assembly diagram.



The CreditRiskAssessmentImport component calls the CreditRiskAssessmentExport component on the assembly diagram of the FoundationServices module. The CreditRiskAssessmentExport component exposes the services of the CreditRiskAssessment rule group.

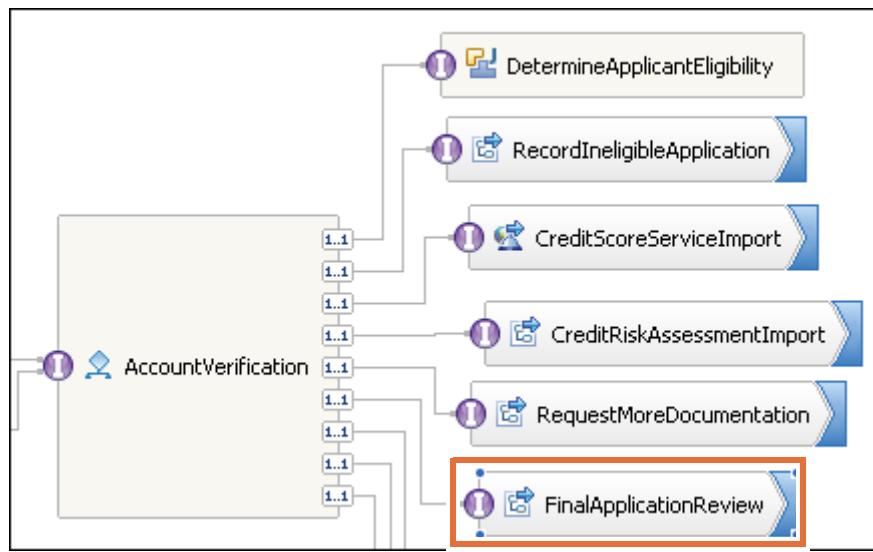


The CreditRiskAssessment rule group contains a rule set that returns a creditRisk value that is based on the value in the creditScore element of the CustomerApplication. Because the creditScore for **ACME** is always 6, the rule set returns a value of MED for the creditRisk field. The rule set updates the contents of the creditRisk element in the CustomerApplication business object with the value that the fired rule returned.

Name	RiskHIGH
Template	CreditRiskTemplate
Presentation	If the customer credit score is greater than <input type="text" value="0"/> and less than <input type="text" value="4"/> then the credit risk is HIGH
Name	RiskMED
Template	CreditRiskTemplate
Presentation	If the customer credit score is greater than <input type="text" value="3"/> and less than <input type="text" value="8"/> then the credit risk is MED
Name	RiskLOW
Template	CreditRiskTemplate
Presentation	If the customer credit score is greater than <input type="text" value="7"/> and less than <input type="text" value="12"/> then the credit risk is LOW

8. When the creditRisk element in the CustomerApplication variable is updated, the creditRisk is evaluated and since the risk is MED, the "Credit risk is MED" link is followed.
9. After the "Credit risk is MED" link is followed, the Final Application Review activity is processed. Final Application Review is an invoke activity that uses the

`FinalApplicationReviewPartner` reference partner to call the `FinalApplicationReview` import component on the `FoundationModule` assembly diagram.



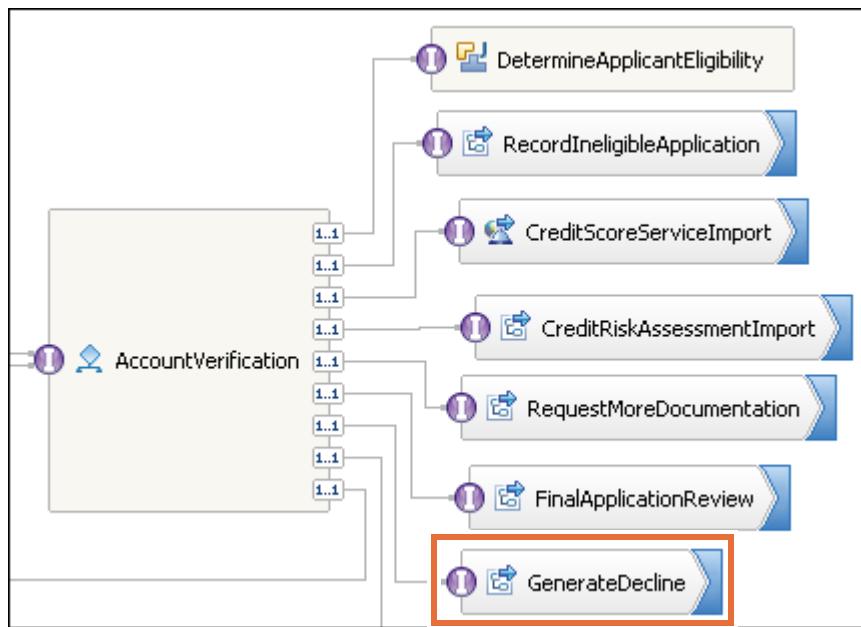
The `FinalApplicationReview` import calls the `FinalApplicationReviewExport` component on the `HumanTaskServices` assembly diagram.



The `FinalApplicationReviewExport` component exposes the services of the `FinalApplicationReview` stand-alone human task. Using a human task user interface such as `AccountProcessingUI`, an employee does a final review of the medium risk application to decide whether the application must be approved or declined. Using the user interface, the employee updates the `applicationDecision` field to `true` (approved) or `false` (declined).

10. Depending on the action that the employee takes, the `applicationDecision` field is examined and either the “Application is declined” link is followed or the “Application is approved” link is followed.
 11. If the “Application is approved” link is followed, the `Create Output` activity is processed. `Create Output` is an assign activity that assigns the `message` element of the `Message` business object to: “Application was approved.”
- If the “Application is declined” link is followed, the `Generate Decline` invoke activity is processed. The `Generate Decline` activity uses the `GenerateDeclinePartner` reference

partner to call the `GenerateDecline` import component on the `FoundationModule` assembly diagram.



The `GenerateDecline` import component calls the `RouteRequestExport` component on the `RouterMediationService` assembly diagram.



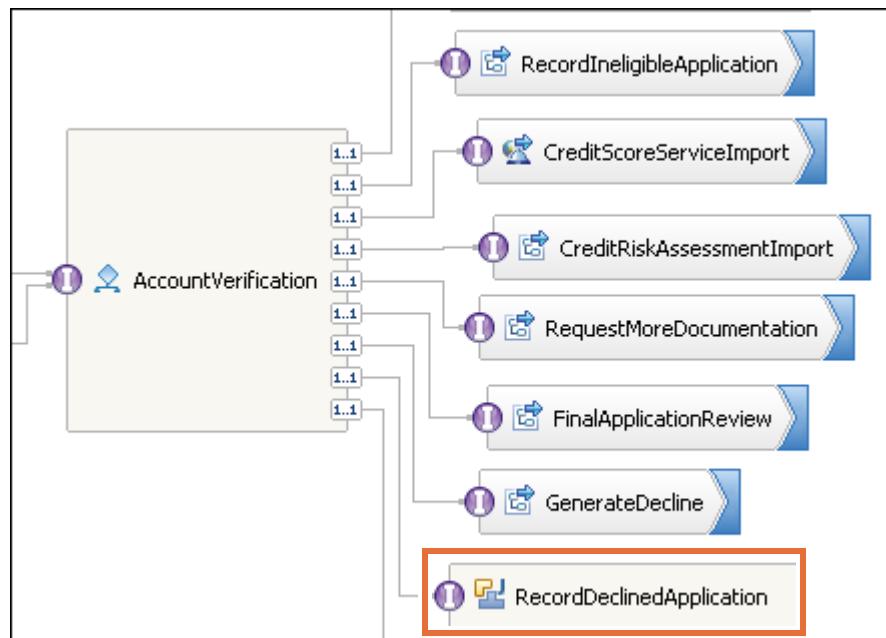
`RouteRequestExport` exposes the services of the `RouteRequest` mediation flow. If the creditRisk is `MED` (which is the case for **ACME**) and applicationDecision is false, the message is routed to the `SpecialDeclineService` import component. The `SpecialDeclineService` import starts the `SpecialDeclineExport` component on the `FoundationServices` assembly diagram.



`SpecialDeclineExport` exposes the services of the `SpecialDecline` Java component. This component sets the `message` element of the `Message` business object to: "Account for customer ACME was routed through special decline because the credit risk was MED."

12. When the `GenerateDecline` activity is processed, the link to `Record Declined Application` is followed. The `Record Declined Application` invoke activity uses the

`RecordDeclinedApplicationPartner` reference partner to call the `RecordDeclinedApplication` Java component on the `FoundationModule` assembly diagram.

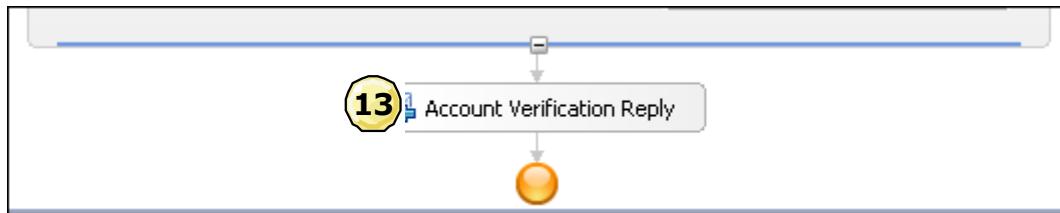


`RecordDeclinedApplication` writes messages to the console and to the server log, indicating that the application was declined.

13. When the message is assigned through the `Create Output` activity, or the `RecordDeclinedApplication` is processed, the `AccountVerification_Flow` is complete, and the process follows the link to `Account Verification Reply`.

`Account Verification Reply` returns one of two messages to the client:

- 1) If the application was approved, a `Message` business object is returned to the client, which contains the `message` element that the `Create Output` assigned: “Application was approved.”
- 2) If the application was denied, a `Message` business object is returned to the client, which contains the `message` element that the `SpecialDecline` service assigned: “Account for customer ACME was routed through special decline because the credit risk was MED.”



End-to-end test

This end-to-end test uses the Business Process Choreographer Explorer client. To test an eligible application with a MED credit risk:

1. In the earlier application test, you opened a browser window to the business user client.

<https://localhost:9443/AccountOpeningUI/Index.jsp>

If this session expired (to check whether it is expired, click **Home** and refresh your browser), you must log in again to create another case. Use `admin` in the **Name** field and `web1sphere` in the **Password** field to log in.

- 2. Create a business case in the business user client.
 - a. Open the browser window that contains the `CreateApplication` business user client interface.
 - b. Under **Business Case**, click **New** to access the `CreateApplication` invocation task.



- ___ c. CreateApplication is the only available task. Click the link.

Business Cases > New

Select a process or task for which you want to create a business case.

▼ Task

→ CreateApplication

- ___ d. On the **CreateApplication** page, in the **Input Data** section, type **ACME** in the `companyName` field. You can leave the remaining fields blank. The `Determine Application Eligibility` Java snippet populates the remaining fields that are based on `companyName`.

Business Cases > New > CreateApplication

Enter the values for the input data and optionally provide additional information to create your task.

▼ Input Data

accountNumber	<input type="text"/>
applicationDate	<input type="text"/>
applicationDecision	<input type="checkbox"/>
comments	<input type="text"/>
companyName	<input type="text" value="ACME"/>



Note

The only available test cases are for the following company names: **AbcCo**, **IBM**, **ACME**, and **TestCo**. Do not deviate from these test cases, or you can receive unpredictable results.

-
- ___ e. Click **Create** at the bottom of the page. The web browser returns to the **Business Cases** page.
- ___ f. Close the browser.

When you submit an application by using `companyName ACME`, the application flows through the process to `Final Application Review`. The `Final Application Review` task is used to update the `applicationDecision` field. If `applicationDecision` is true, the application flows through **Create Output > Account Verification Reply**. If `applicationDecision` is false, the application flows through **Generate Decline** (this time through `SpecialDeclineService`) > **Record Declined Application > Create Output > Account Verification Reply**.

- 3. When you submit an application by using `companyName ACME`, the application flows through the process to `Final Application Review`. Using the Business Process Choreographer Explorer client, change the `applicationDescition` field to `true` to complete the process.
 - a. In IBM Integration Designer, switch to the Business Process Choreographer Explorer tab. If you logged out or the session is timed out, then use the instruction that was provided earlier to log in to the Business Process Choreographer. Use `admin` in the **User Name** field and `web1sphere` in the **Password** field to log back in.
 - b. The default page is the **My To-dos** page. A task is waiting named `Final Application Review`.

My To-dos

Use this page to work on tasks that are assigned to you. i

<input type="checkbox"/>	Priority	Task Name	State	Kind	Owner	Origin
<input type="checkbox"/>	5	Final Application Review	Ready	To-do Task	admin	

Items found: 1 Items selected: 0 << Page 1 of 1 >>

- c. Place a check next to the task to select it, and click **Work on**.
- d. On the **Task Message** page, scroll to the **Task Output Message** section and click **Edit Source** (at the bottom of the section).
- e. In Windows Explorer, browse to the directory where the support files for this exercise are stored (`C:\labfiles\Support Files\EX1`).
- f. Open `EX1_Test_Data3.txt` in a text editor such as Notepad (note the file extension `.txt`).

- __ g. Copy the text in `EX1_Test_Data3.txt` and paste it over the existing text in the **Task Output Message Source View** window.

Source View*

```
<?xml version="1.0" encoding="UTF-8"?>
<p:InputCriterionResponse xsi:type="p:InputCriterionResponse_.type"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:p="http://FoundationLibrary/FinalApplicationReview">
<Output>
<accountNumber>ACM002</accountNumber>
<applicationDate>Oct 02, 2014</applicationDate>
<applicationDecision>false</applicationDecision>
<comments>None</comments>
<companyName>ACME</companyName>
<contactFirstName>Torsten</contactFirstName>
<contactLastName>Frings</contactLastName>
<contactPhoneNumber>905-555-7234</contactPhoneNumber>
<creditRating>C+</creditRating>
<creditReportNeeded>true</creditReportNeeded>
<creditRisk>MED</creditRisk>
<creditScore>6</creditScore>
<customerCity>Berlin</customerCity>
<customerCountry>Germany</customerCountry>
```

Cancel **Validate** **Confirm** **Reset**

- __ h. Click **Confirm**.
- __ i. When you are returned to the **Form View**, verify that the `applicationDecision` field in the **Task Output Message** area is cleared.
- __ j. At the top of the page, click **Complete** to finish the task.
- __ k. Leave Windows Explorer open and close `EX1_Test_Data3.txt`.
- __ 4. Verify the path that the application took through the remainder of the business process by examining the messages in the **Server Logs** view.
- __ a. In IBM Integration Designer, switch to the **Server Logs** view.

- ___ b. The following messages in the server log confirm a successful path through the business process.

Generate Decline Special - begins

Generate Decline Special - Account for customer ACME was routed through special decline because the credit risk was MED

Generate Decline Special- ends

Record Declined Application - begins

Record Declined Application - ends

Contents

[Java] Generate Decline Special - Account for customer ACME was routed through special decline because the credit risk was MED
[Java] Generate Decline Special - ends
[Java] Record Declined Application - begins
[Java] Record Declined Application - ends



Note

If you cannot read the log message contents, double-click the message.



Optional

If time allows, run the test again with `applicationDecision` set to `true`. For this test case, `Special Decline` is not called, so the `Special Decline` messages are not shown in the **Server Logs** view.

- ___ 5. Click the **Logout** link to log out of the Business Process Choreographer Explorer.
- ___ 6. Close the Business Process Choreographer Explorer tab.
- ___ 7. Remove the applications from the server.
 - ___ a. In the **Servers** view, right-click **IBM Process Server V8.5.7** and choose **Add and Remove** from the menu.
 - ___ b. Click **Remove All** and click **Finish**.
- ___ 8. (Optional) Stop IBM Process Server. IBM Process Center Server is required to test the next part of this exercise, not IBM Process Server. You might also leave IBM Process Server running throughout this exercise, but be aware that running two WebSphere Application Server profiles concurrently places a tremendous strain on system resources. If you choose to leave the IBM Process Server running, the server performance can be affected.
 - ___ a. To stop IBM Process Server, in the **Servers** view, right-click **IBM Process Server v8.5.7 at localhost**.
 - ___ b. Choose **Stop** from the menu.
- ___ 9. Close the Foundation module assembly diagram editor.
- ___ 10. Do not close IBM Integration Designer. You can minimize it.

1.2.Create a business process diagram in IBM Process Designer

In the previous walk-through, arbitrary Java code determined the eligibility of a customer account. The Java code is written in the `Determine Applicant Eligibility` component. This code populates the request with sample data, which depends upon the value of the customer company name.

In this example, you replace this Java component with a business process built-in IBM Process Designer. You still use sample data, but the data is extracted from a business process diagram, not from a Java SCA component.

Part 1: Associate a library with the repository

The development team created a library that contains reusable, shareable artifacts for developing business models. This library is called `FoundationLibrary`. In this portion of the exercise, you associate this library with a toolkit in the repository to help develop models.

IBM Process Designer maintains its business process definition (BPD) artifacts in the centralized IBM Process Center repository. To create, view, edit, or inspect a BPD in IBM Process Designer, you must first start the IBM Process Center environment.

- 1. Start the Process Center deployment manager.
 - a. On your Windows desktop, select the shortcut that is titled: **Start Process Center deployment manager**. Double-click or press Enter to open the shortcut. It takes several minutes for deployment manager to start. When the deployment manager starts, you are prompted to press any key to continue. Press any key to close the command window.
- 2. Start the Process Center node agent.
 - a. On your Windows desktop, select the shortcut that is titled: **Start Process Center node agent**. Double-click or press Enter to open the shortcut. It takes several minutes for node agent to start. When the node agent starts, you are prompted to press any key to continue. Press any key to close the command window.
- 3. Start the Process Center Cluster.
 - a. On your Windows desktop, select the shortcut that is titled **Start Process Center Cluster**. Double-click or press Enter to open the shortcut. It takes several minutes (longer than the deployment manager and node agent processes) for the cluster to start. When the cluster starts, you are prompted to press any key to continue. Press any key to close the command window.



Hint

You can also start IBM Process Center environment from a DOS command window. If you open a DOS command window, browse to the following directory:

C:\IBM\BPM\v8.5\profiles\dmgrProfile\bin

- Type the following command to start the IBM Process Center Deployment Manager:

```
startManager.bat
```

- To start the node agent and the cluster, browse to the following directory:

```
C:\IBM\BPM\v8.5\profiles\Node1Profile\bin
```

- Type the following command to start the IBM Process Center node agent:

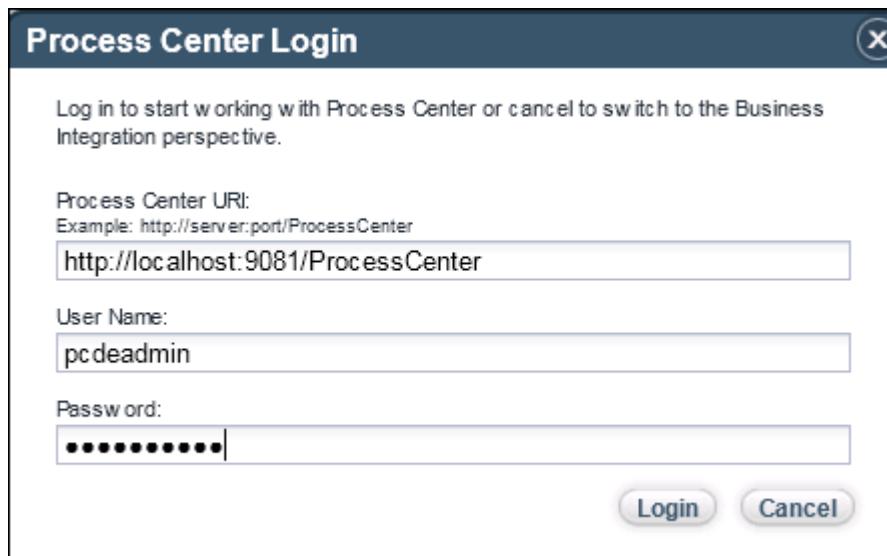
```
startNode.bat
```

- Type the following command to start the IBM Process Center Single Cluster:

```
startServer.bat SingleClusterMember1
```

When installed, IBM Process Center installs and configures several databases across a single instance to form the IBM Process Center repository. In the example of this course, the name of that instance is `BPMINST`. Several databases are installed in that instance.

- 4. After IBM Process Center environment starts, return to IBM Integration Designer.
- 5. Open the Process Center perspective. Click **Window > Switch to Process Center**.
- 6. Log in to the Process Center perspective by using the following connection information:
 - Process Center URL: `http://localhost:9081/ProcessCenter`
 - User name: `pcdeadmin`
 - Password: `web1sphere`



- 7. Click **Cancel** in the **Secure Storage** window. Multiple different windows are displayed. It is okay if the display order of these windows is different from the order that is listed in the following steps.
- 8. Click **Yes** when prompted with Security Alerts messages.
- 9. Click **Cancel** again in the **Secure Storage** window.
- 10. Click **OK** in the **Secure Storage Warning** window.
- 11. Close the **Getting Started** window.

- ___ 12. Click the **Process Apps** tab. This tab contains all the process applications in the repository. A process application contains deployable process resources.
- ___ 13. Click the **Open in workspace** link that is associated with the **Account Management (AM_101)** process app. This action adds the process application from the repository into the IBM Integration Designer workspace.

The screenshot shows the 'Process Apps' tab selected in the top navigation bar. Below it is a search bar with 'Sort By: Recently Updated' and a dropdown menu. The main area lists several process applications:

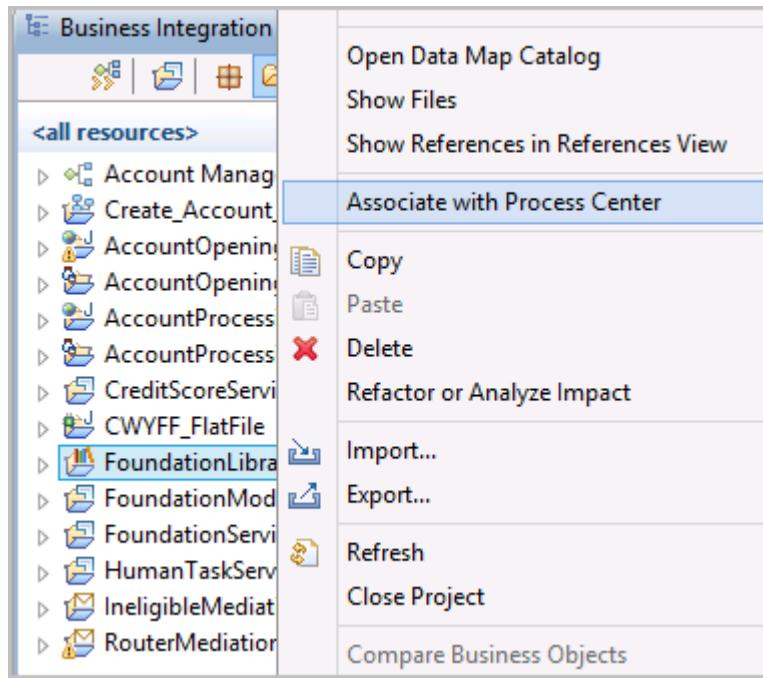
- Basic Governance for Install Sample (BGIS)** (Last updated on 9/27/16 by pdeadmin) with an 'Open in workspace' link.
- Account Management (AM_1012)** (Last updated on 9/27/16 by pdeadmin) with an 'Open in workspace' link.
- AIS Fault Handling (AISF)** (Last updated on 9/27/16 by pdeadmin) with an 'Open in workspace' link.
- Account Management (AM_101)** (Last updated on 9/27/16 by pdeadmin) highlighted with a red box, and its 'Open in workspace' link is also highlighted with a red box.
- Hiring Sample (HSS)** (Last updated on 4/9/16 by pdeadmin) with an 'Open in workspace' link.
- Hiring Sample Advanced (HSBV1)** (Last updated on 4/9/16 by pdeadmin) with an 'Open in workspace' link.



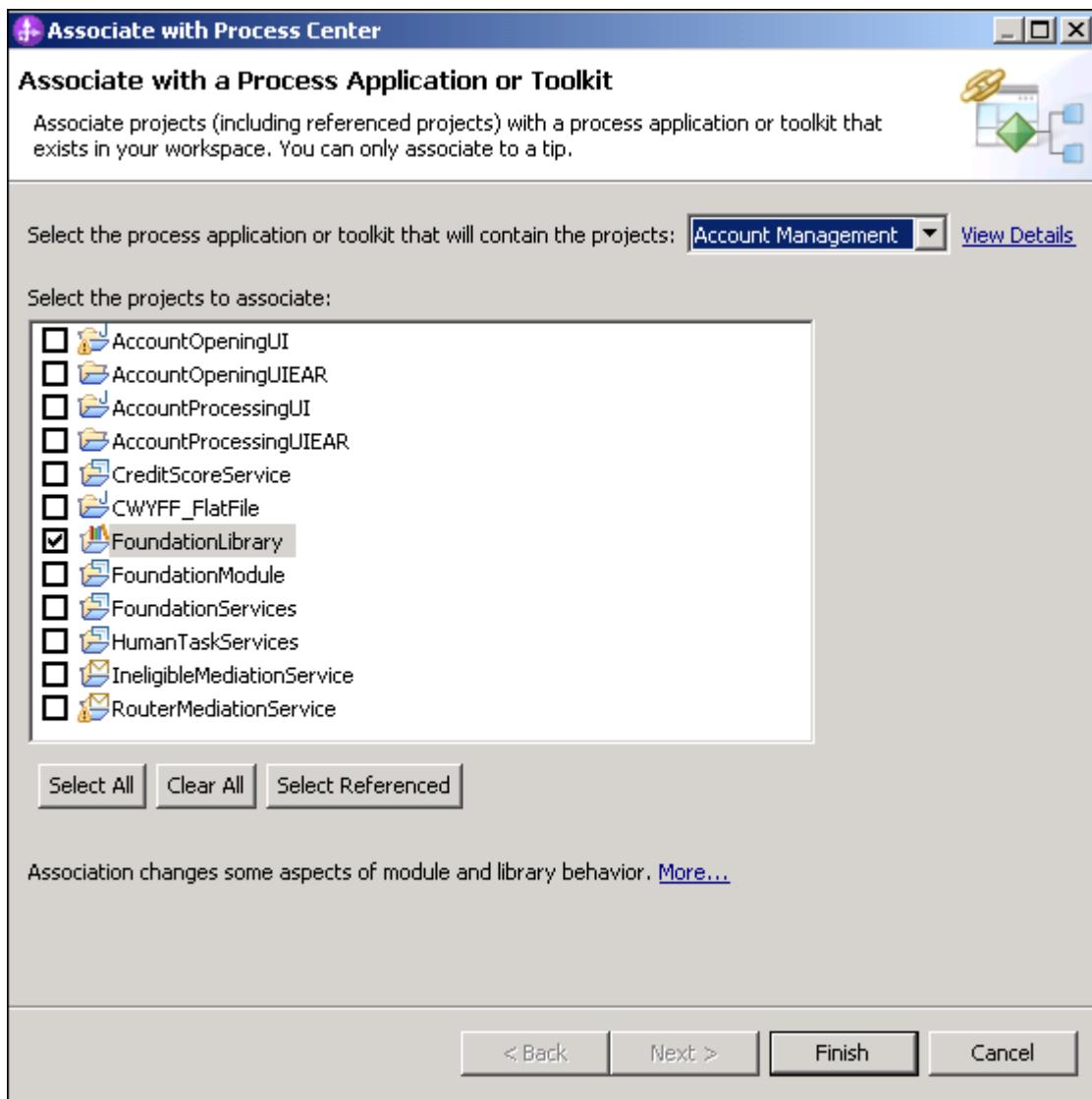
Make sure that you click **Open in workspace** against **Account Management** with **AM_101** (not AM_1012) for this lab.

- ___ 14. If you are prompted with the **Tip** dialog box, close it by clicking **X**.

- 15. Associate the **FoundationLibrary** with the **Account Management** process app in IBM Process Center.
- a. **FoundationLibrary** contains reusable library resources, such as data objects and interfaces. Right-click **FoundationLibrary** and click **Associate with Process Center**.

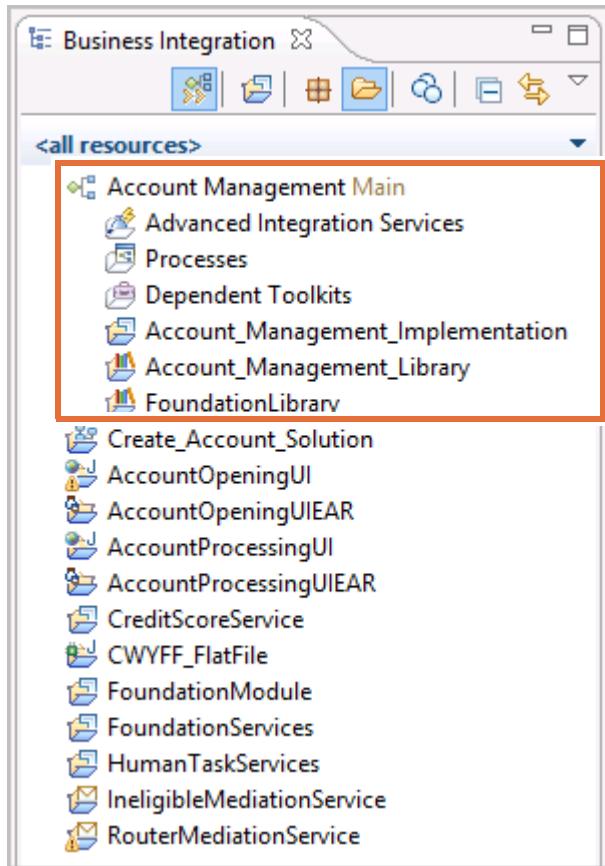


- ___ b. In the association window, you must select the process application or toolkit with which the projects are associated, and the projects to associate. Ensure that the selected process application is **Account Management**, and make sure the project to associate is the **FoundationLibrary**.



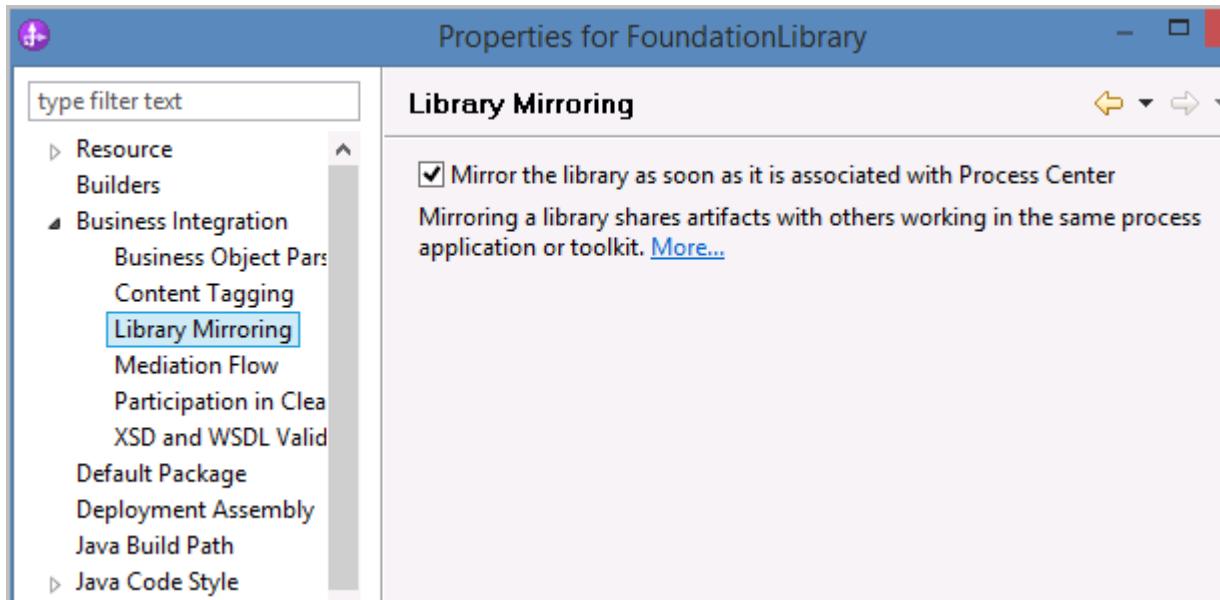
- ___ c. Click **Finish**.
 ___ d. When prompted to switch to **Detailed mode**, click **Yes**.
 ___ e. Wait for the tool to associate the IBM Integration Designer project with the toolkit. It can take a few minutes for the association with the Process Center to take place.

- ___ f. Examine the **Account Management Main** project on the left. After the association is complete, the **FoundationLibrary** is shown as an element inside the **Account Management** process app.

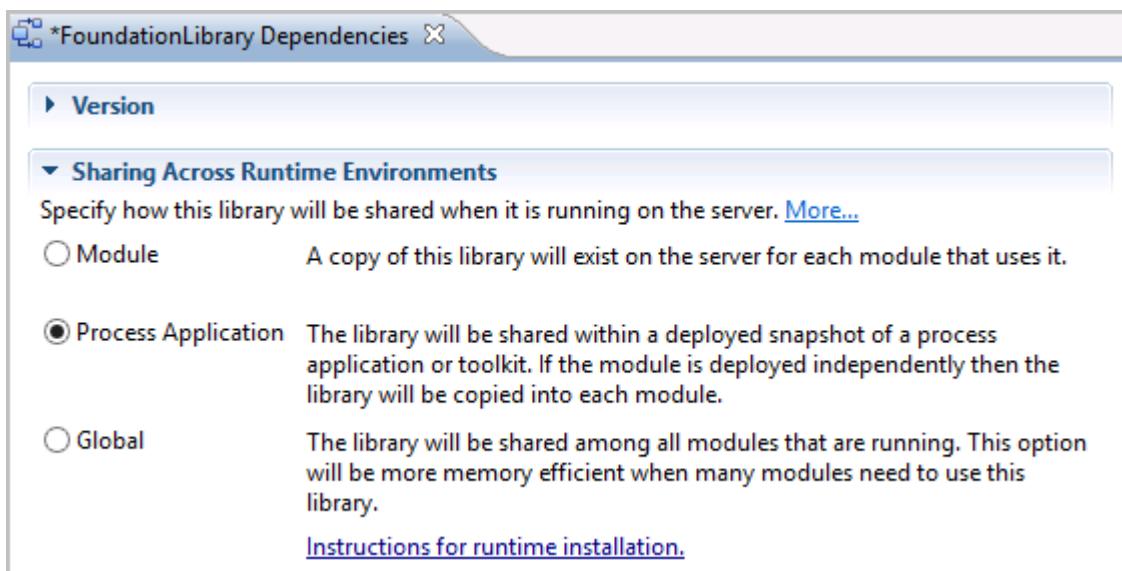


- ___ 16. Assets from this library must be shared with developers who are building process applications. This phenomenon is known as “Library mirroring”. Enable library mirroring for the **FoundationLibrary**.
- ___ a. Right-click **FoundationLibrary** and click **Properties**.
 - ___ b. Expand **Business Integration** and select **Library Mirroring**.

- __ c. Select **Mirror the library as soon as it is associated with Process Center**.



- __ d. Click **OK** to close the **Properties** window.
- 17. Whenever a library in IBM Integration Designer is shared, you must specify how the library is shared. The library can be shared in an IBM Integration Designer module, in an IBM Process Center process application, or globally, among all modules that are running. For the purposes of this exercise, set the `FoundationLibrary` so that it can be shared at the process application level.
- __ a. Expand the `FoundationLibrary` and double-click **Dependencies**.
 - __ b. Expand the section that is named **Sharing Across Runtime Environments**.
 - __ c. Select the **Process Application** option.

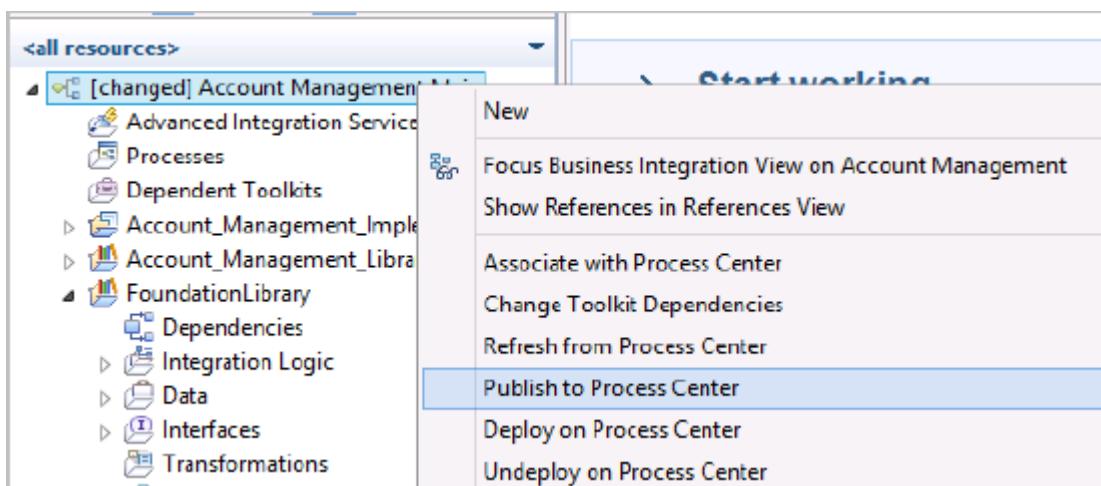


- __ d. Save your changes and close the editor.

**Note**

In IBM Integration Designer, libraries are shared at the module level by default. Libraries can be associated with either process apps or toolkits (containers). When a library is associated with a container from IBM Process Designer, the sharing properties of the library must be changed. Either the **Process Application** level or the **Global** level is acceptable for sharing properties. When you share a library at the global level, you must create variables in WebSphere Application Server.

- 18. After changes are made to any resource in a repository project, those changes must be published back to the repository. The changes must be published to share the mirrored resources from the **FoundationLibrary** back to the repository.
 - a. Right-click **Account Management Main** and click **Publish to Process Center**.



- b. Click **OK** in the **Library Mirroring** window. It can take several moments to publish the changes to the repository. If you are prompted about publishing XSD resources to the repository, click **OK**.

**Hint**

When library mirroring is active, and data objects are published to the repository, the XSD files that make up a data object can be edited in IBM Process Designer. If these resources are changed, the data types might not be compatible with the projects in IBM Integration Designer anymore.

Whenever you publish a mirrored library to the repository, consider setting permissions on the resources in IBM Process Designer. These security permissions can control or prevent users from inadvertently editing data objects, which might lead to data incompatibility.

The final step in this section is to take a snapshot of the **Account Management** process app, so that you can use the contents, which now include the **FoundationLibrary** mirrored artifacts, in process applications.

- 19. On the Windows desktop, locate the icon that is labeled **Start IBM Process Designer**. Double-click the icon or press **Enter** to start IBM Process Designer.

- ___ 20. As the splash screen for IBM Process Designer begins loading, you are prompted to enter a user name and password to connect to the IBM Process Center repository. Enter `pdeadmin` in the **User name** field and `web1sphere` in the **Password** field.
- ___ 21. Click **Log In**. After a few moments, IBM Process Designer starts.
- ___ 22. When prompted with security alerts, click **Yes**.
- ___ 23. If you do not see the **Process Apps** tab, you might be in the Designer view. In that case, click **Process Center** on the upper-right corner of the screen. Otherwise, if the **Process Apps** tab is listed, then skip to the next step.
- ___ 24. Click the **Process Apps** tab.
- ___ 25. Examine the artifacts of the **Account Management (AM_101)** process application.
 - ___ a. Locate the process app named **Account Management (AM_101)**.
 - ___ b. Click the **Open in Designer** link that is associated with the process app. The process app opens in Designer view.

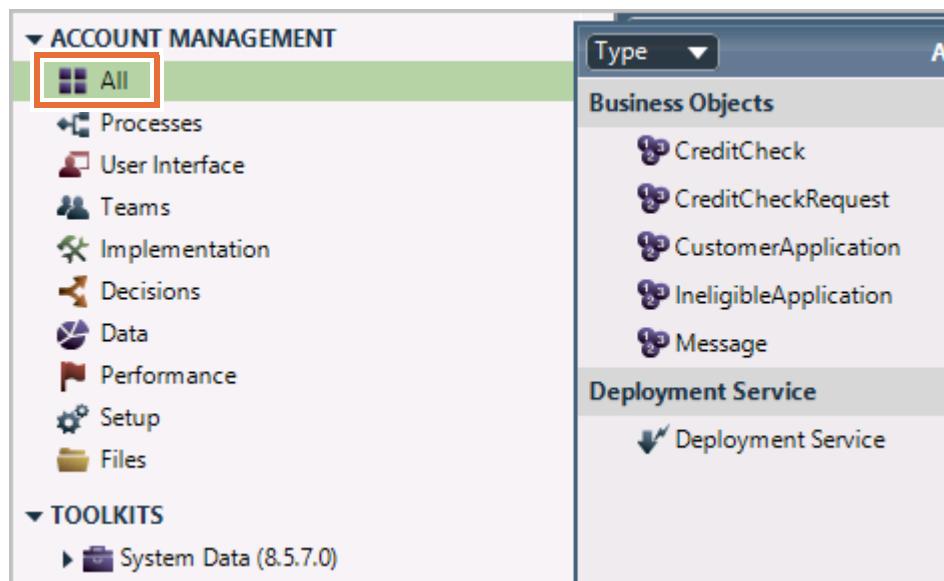
Sort By: Recently Updated All | Favorites

	Basic Governance for Install Sample (BGIS)	
	Account Management (AM_1012)	
	AIS Fault Handling (AISF)	
	Account Management (AM_101)	
	Hiring Sample (HSS)	

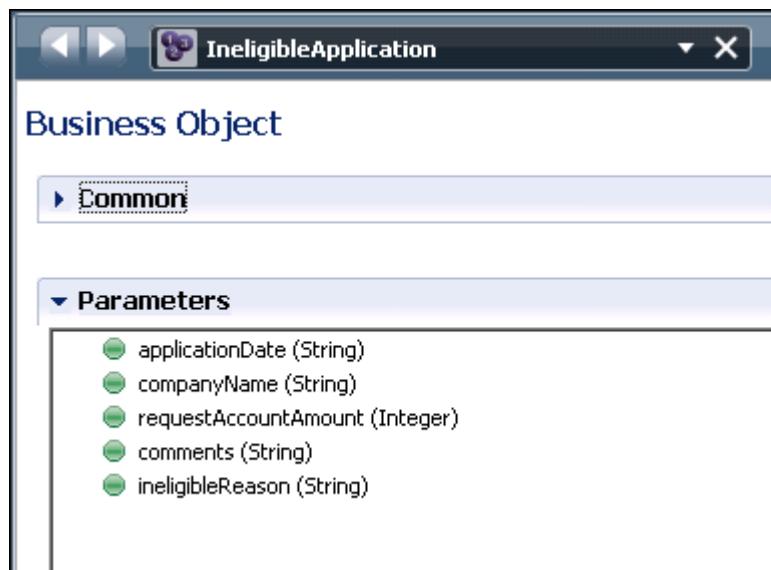


Make sure that you click **Open in Designer** against **Account Management** with **AM_101** (not **AM_1012**) for this lab.

- c. Select **All** from the list of available artifacts. Interfaces do not translate from IBM Integration Designer to IBM Process Designer. The only usable artifacts from the library are business objects.



- d. Double-click the `IneligibleApplication` business object. It opens in the editor.
 — e. Examine the parameters of the `IneligibleApplication` business object. When a customer application is ineligible, you can store comments and a reason for the ineligibility.

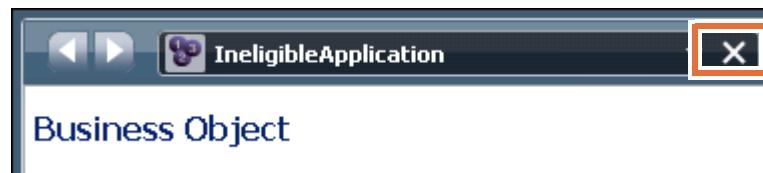


**Hint**

The XSD files that make up a data object can be edited in the editor for IBM Process Designer. If these resources are changed, the data types might not be compatible with the projects in IBM Integration Designer any longer.

Whenever you publish a mirrored library to the repository, consider setting permissions on the resources. These security permissions can control or prevent users from inadvertently editing data objects, which might lead to data incompatibility.

- f. Click the **X** to close the business object editor.



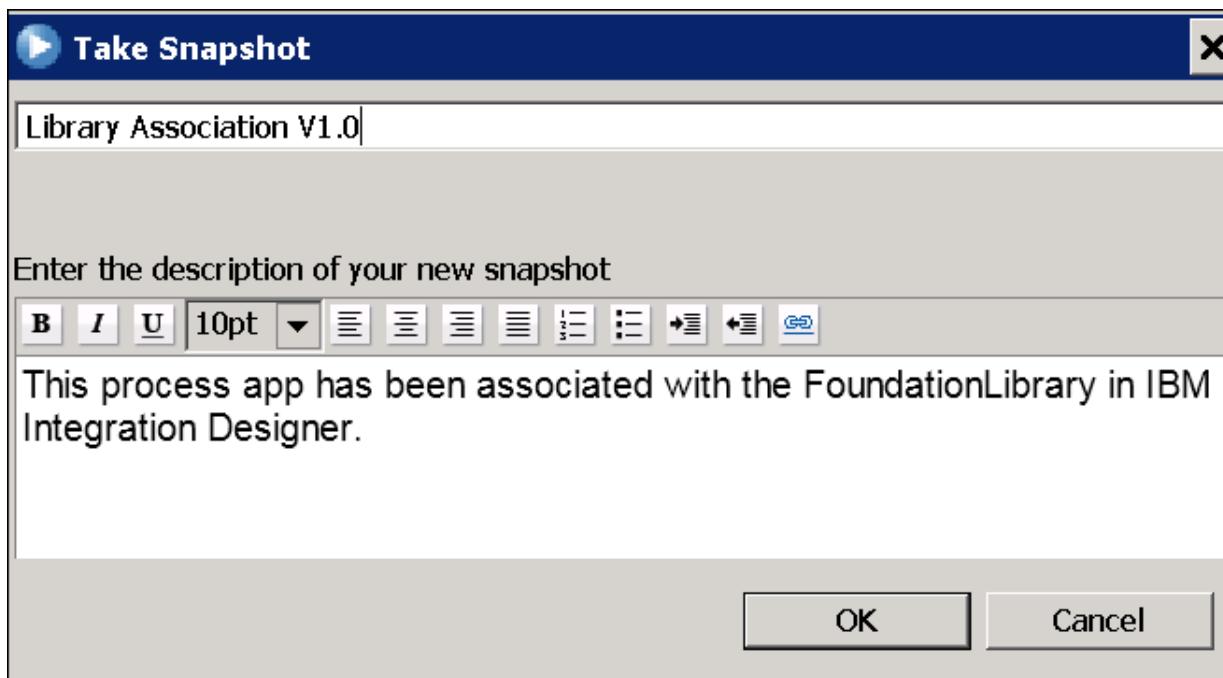
26. Take a snapshot.

- a. Click **Snapshot** in the upper right of the Designer view.



- b. Set the following values for the snapshot:

- Name: Library Association V1.0
- Description: This process app has been associated with the FoundationLibrary in IBM Integration Designer.



- c. Click **OK**.
- d. The snapshot is available in the revision history.



Part 2: Create a business process diagram

Development artifacts, which were created in IBM Integration Designer, are ready to be used in models for IBM Process Designer. In this section of the exercise, you create a business process to contain the `DetermineApplicantEligibility` logic. This process determines whether a customer application is eligible based on the company name.

In a practical application of IBM Process Designer, you can model a process that uses a customer information file or another source of data to form a decision.

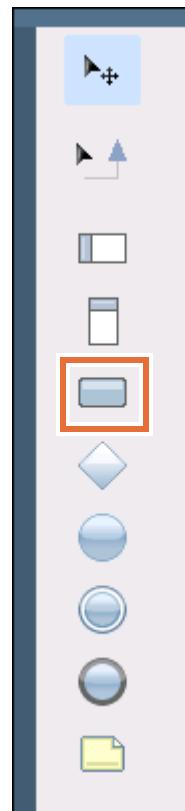
- 1. In the Designer view of the `Account Management` process app, click the plus (+) sign next to **Processes**.



- 2. Click **Business Process Definition**.
- 3. Set the **Name** of the new business process definition to `Determine Applicant Eligibility`.
- 4. Click **Finish**. The business process editor opens. A `start` event and an `end` event are added automatically.

In a real-world scenario, this process likely does activities such as retrieving the customer profile, assessing the credit record, and reviewing the customer contract. You can build the process with these activities, but you are not creating any implementations in this course to support these functions.

- ___ 5. Add the first activity, named **Retrieve the Customer Profile**, to the process diagram.
- ___ a. Click the **Activity** icon in the palette.

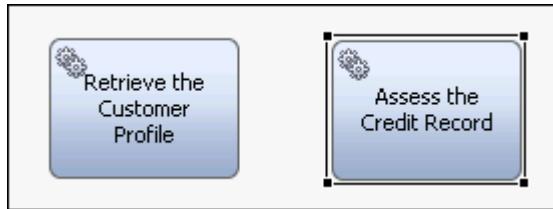


- ___ b. Click anywhere in the **System** lane to add the activity to the diagram. The default name for the activity is **Untitled**.
- ___ c. Select the **Properties** tab.
- ___ d. Select the **General** option in **Properties**.
- ___ e. Under the set of **Common** properties, select the **Name** field. Change the name from Untitled to: **Retrieve the Customer Profile**

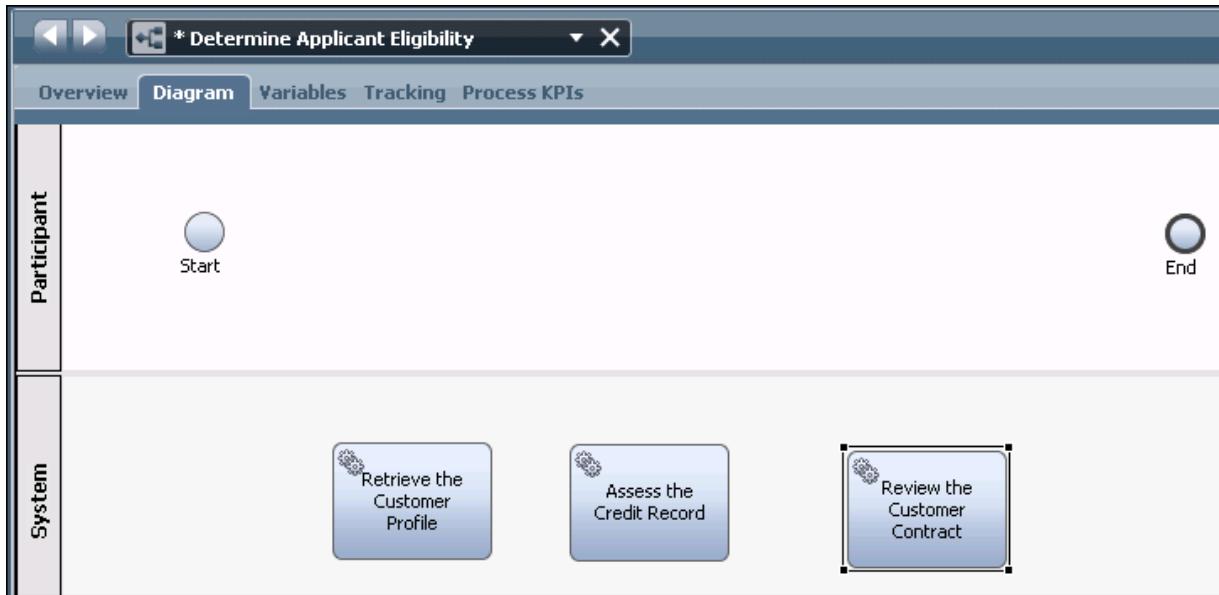
Properties		Validation Errors	Where Used
General		Common	
Simulation	Name: <input type="text" value="Retrieve the Customer Profile"/>		
Implementation	Presentation: <input checked="" type="radio"/> Color <input type="radio"/> Icon		
Assignments	Presentation Color: <input type="text" value="Default"/>		
Data Mapping			
Pre & Post	<i>Click Edit to add or edit text.</i>		

If you want, you can change the **Presentation Color** as well.

- 6. Add a second activity that is named **Assess the Credit Record** next to the retrieval activity.
 - a. Click the **Activity** icon in the palette.
 - b. To add the activity to the diagram, click anywhere in the **System** lane to the right of the retrieval activity.
 - c. In the **Properties** tab, set the name of the activity to: **Assess the Credit Record**



- d. If you want, you can change the **Presentation Color** as well.
- 7. Add the final activity, named **Review the Customer Contract**, in the **System** lane of the process diagram. Follow the preceding steps to add the activity.

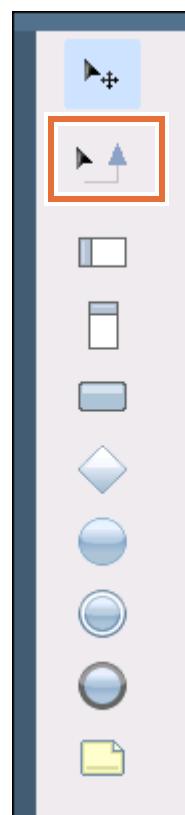


Questions

Why was the third activity not added to the Participant lane?

If this example was an actual business process, it is likely that the first two activities (the retrieval and assessment) would be automated, or **System** activities. Therefore, reviewing the customer contract is likely a human task, and requires intervention from a user participant. However, this process is automated for the purposes of this lab.

- 8. Connect the activities.
 - a. Click the **Sequence Flow** connection tool in the palette.

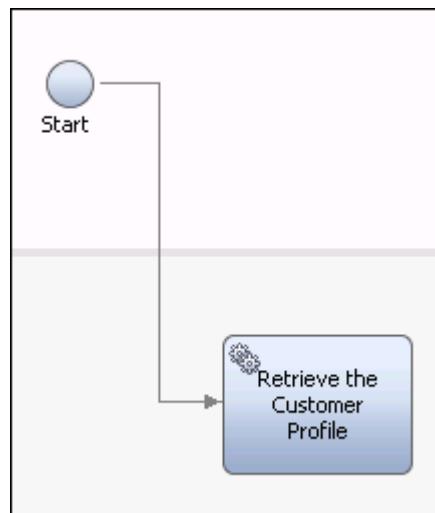


- b. Hover over the **Start** element. The connection tool shows the available connection points and a connection icon.

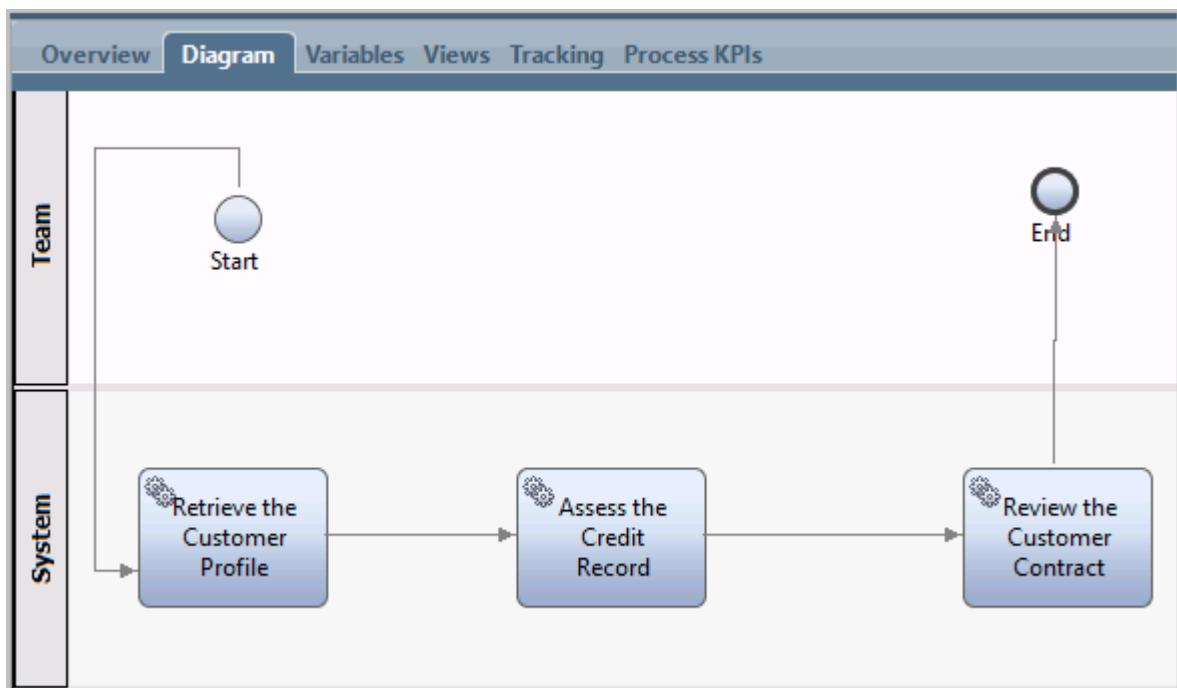


- c. Click the **Start** element. This element starts the connection.

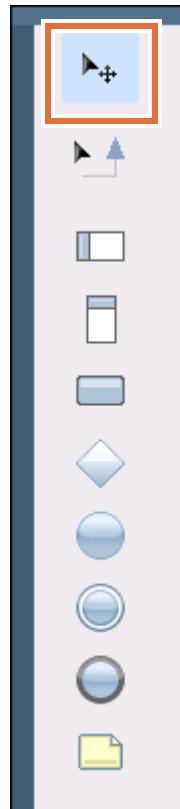
- ___ d. Click the **Retrieve the Customer Profile** activity. The **Start** element is connected to the retrieval activity, indicating the direction of flow control.



- ___ e. Connect the retrieval activity to **Assess the Credit Record**.
___ f. Connect the assessment activity to **Review the Customer Contract**.
___ g. Connect the review activity to the **End** element.



- __ h. Select the **Selection Tool** from the palette.

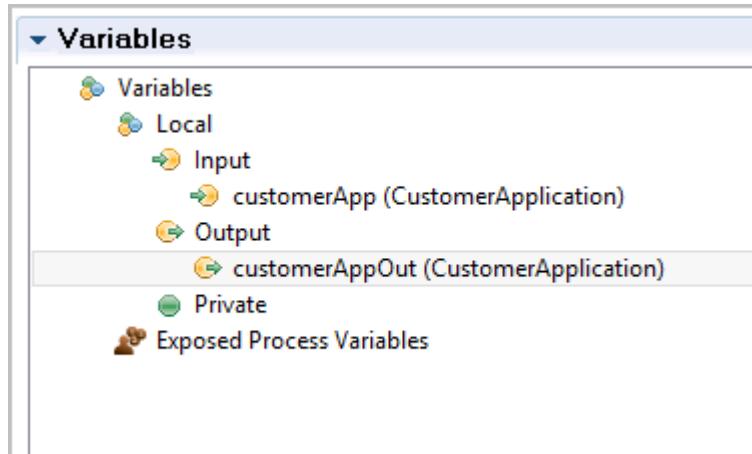


- __ i. Save the diagram.
- __ 9. The business process diagram takes a customer's application as input and returns the application with the customer eligibility as output. In this step, set the input and output variables for the business process diagram.
- __ a. Click the **Variables** tab.
 - __ b. Click **Add Input**.
 - __ c. In the **Details** section, set the **Name** of the input variable to: `customerApp`
 - __ d. Click **Select**, which is next to the **Variable Type** field.
 - __ e. Click **CustomerApplication**.

▼ Details	
Name:	<input type="text" value="customerApp"/>
Documentation: (Edit)	<p>Click Edit to add or edit text.</p>
Is List:	<input type="checkbox"/>
Variable Type:	<input type="button" value="Select..."/> <input type="button" value="New ..."/>

- __ f. Click **Add Output**.
- __ g. In the **Details** section, set the **Name** of the output variable to: `customerAppOut`

- __ h. Click **Select**, which is next to the **Variable Type** field.
- __ i. Click **CustomerApplication**. You now have an input and an output variable for the business process diagram.



- __ j. Save your work.

Part 3: Create an implementation for a business process activity

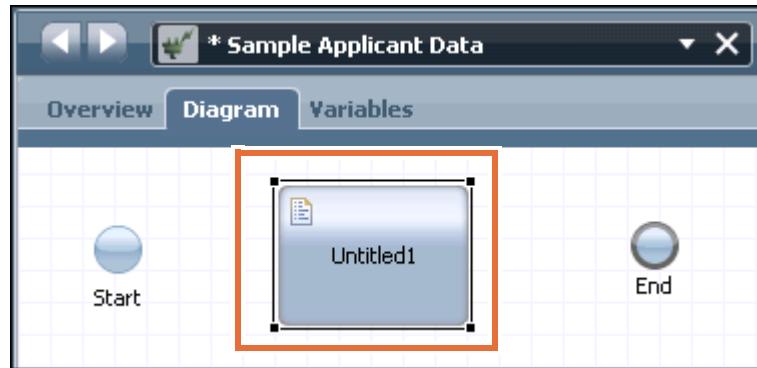
In a real-world scenario, you must build an implementation for each of the activities in your business process diagram. It is not practical to attempt to build this code in this exercise, so your implementation uses sample data. This activity is similar to the Java component in the assembly diagram.

- __ 1. Add a sample implementation to the retrieval activity.
 - __ a. Click the **Diagram** tab.
 - __ b. Select **Selection Tool** from the palette.
 - __ c. Click the **Retrieve the Customer Profile** activity from the diagram to select it.
 - __ d. In the **Properties** view, click the **Implementation** tab.
 - __ e. Verify that the implementation is set to a **System Task**.
 - __ f. To add an implementation to this activity, click **New**.

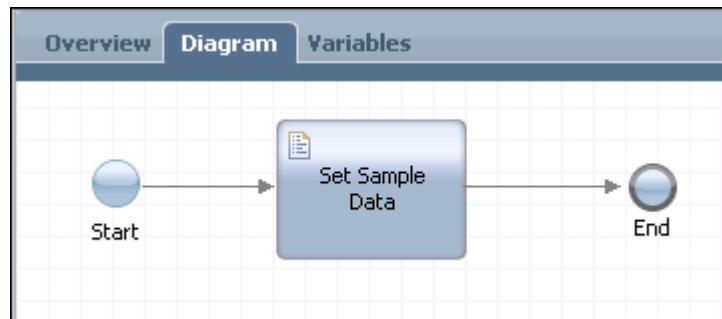


- __ g. Set the name of the service to **Sample Applicant Data** and click **Finish**.
- __ 2. Add a **Server Script** activity to the implementation editor.
 - __ a. Click the **Server Script** activity.

- ___ b. Add the **Server Script** between the start and end events.



- ___ c. In the **Properties** view, select the **Step** tab.
 ___ d. Change the **Name** of the activity to: Set Sample Data
 ___ 3. Connect the server script to the diagram.
 ___ a. Click the **Sequence Flow** connection tool from the palette.
 ___ b. Connect the Start event to the server script activity.
 ___ c. Connect the server script activity to the End event.

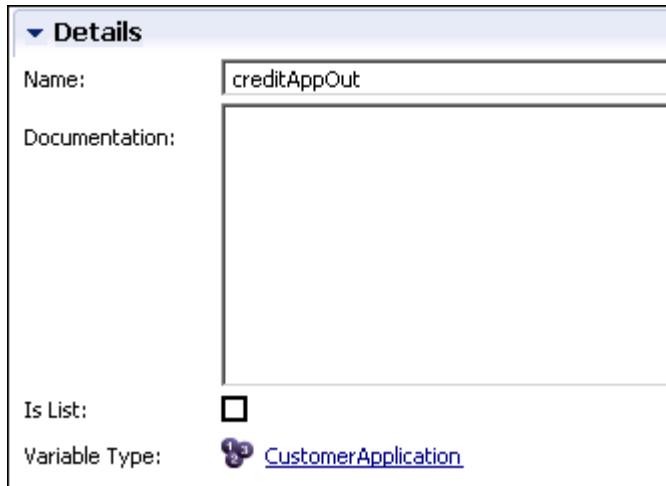


- ___ d. Click the **Selection Tool** from the palette.
 ___ 4. Set the variables for the implementation.
 ___ a. Click the **Variables** tab for the Sample Applicant Data.
 ___ b. Click **Add Input**.
 ___ c. Set the **Name** of the input variable to: creditAppIn
 ___ d. Set the **Variable Type** to: CustomerApplication

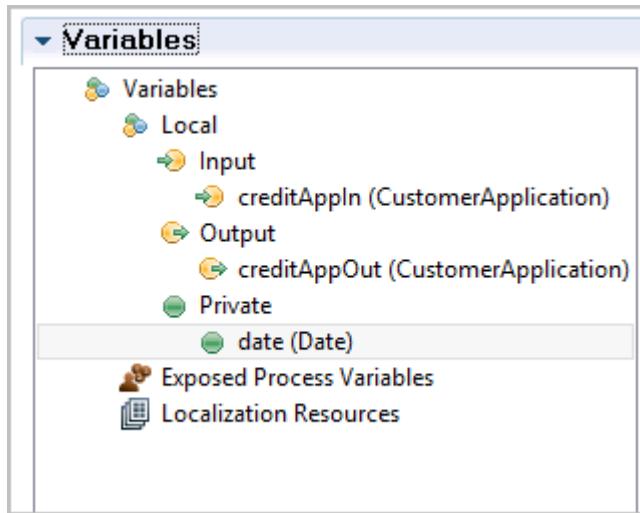
▼ Details	
Name:	creditAppIn
Documentation:	
Is List:	<input type="checkbox"/>
Variable Type:	CustomerApplication

- ___ e. Click **Add Output**.

- ___ f. Set the **Name** of the output variable to: creditAppOut
- ___ g. Set the **Variable Type** to: CustomerApplication

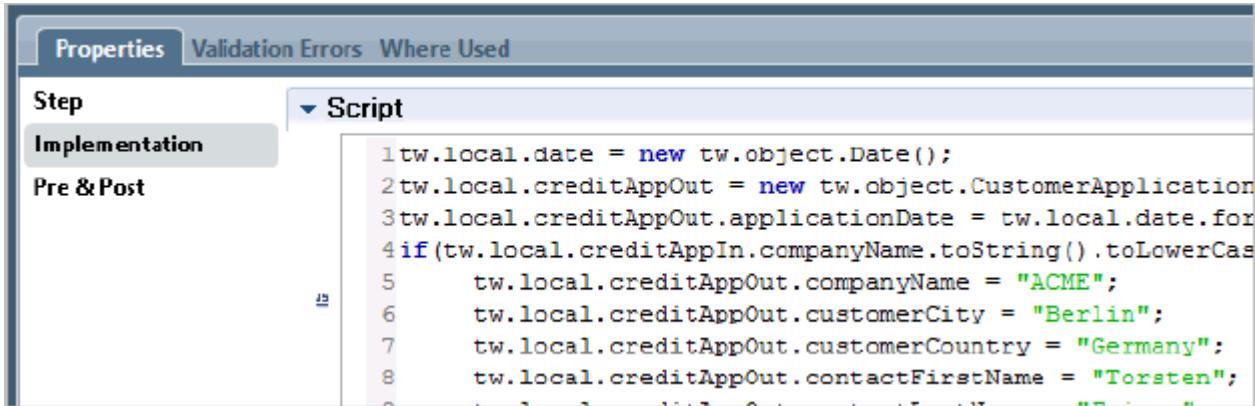


- ___ h. Click **Add Private**.
- ___ i. Set the **Name** of the private variable to: date
- ___ j. Set the **Variable Type** to: Date



- ___ k. Click the **Diagram** tab to return to the implementation diagram.
5. Add the sample data to the implementation of the server script.
- ___ a. Open Windows Explorer and browse to the C:\labfiles\Support Files\EX1 directory.
 - ___ b. Open the file titled `SampleApplicantData.txt` in a text editor, such as Notepad.
 - ___ c. Copy the entire contents of this file.
 - ___ d. In IBM Process Designer, click the server script activity (Set Sample Data) to select it.
 - ___ e. In the **Properties** tab, click the **Implementation** option.

- ___ f. Paste the contents of the sample file into the **Script** field.



```

1 tw.local.date = new tw.object.Date();
2 tw.local.creditAppOut = new tw.object.CustomerApplication
3 tw.local.creditAppOut.applicationDate = tw.local.date.for
4 if(tw.local.creditAppIn.companyName.toString().toLowerCase()
5     tw.local.creditAppOut.companyName = "ACME";
6     tw.local.creditAppOut.customerCity = "Berlin";
7     tw.local.creditAppOut.customerCountry = "Germany";
8     tw.local.creditAppOut.contactFirstName = "Torsten";
9

```

- ___ g. Save your work.
- ___ h. Close the editor for Sample Applicant Data.
- ___ 6. Set the implementations on the other activities to default system services.
- Select the **Assess the Credit Record** activity from the Determine Applicant Eligibility business process diagram.
 - Select the **Implementation** tab in the **Properties** view.
 - Verify that the implementation is set to **System Task** and the task uses the **Default System Service**.



Information

The **Default System Service** is a system service that passes data from the start element to the end element with no activities between. If you want, you can click the **Default System Service** link to examine the service.



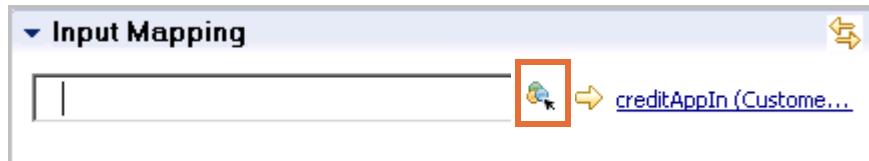
Implementation

System Task ↘ Default System Service

Delete task on completion?

- Repeat these steps for the **Review the Customer Contract** activity.
- ___ 7. Map the data from the process to the implementation.
- In the Determine Applicant Eligibility editor, select the **Retrieve the Customer Profile** activity.
 - In the **Properties** view at the bottom, select the **Data Mapping** tab. This option copies data from the parent business process into the **Sample Applicant Data** implementation.

- ___ c. Under the **Input Mapping** section, click the variable icon to choose the variable to set.



- ___ d. Double-click `customerApp` (`CustomerApplication`). The local variable `customerApp`, which you set as an input variable in a previous step, is mapped to `creditAppIn`.



- ___ e. Follow the preceding steps to set the output mapping such that `creditAppOut` maps to `customerAppOut`.

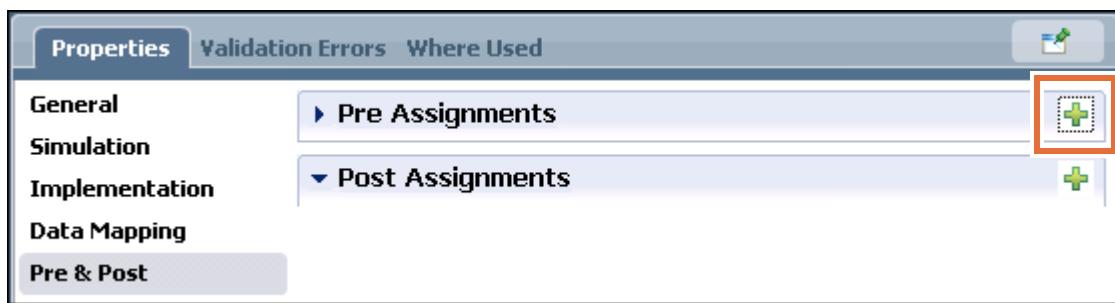


- ___ f. Save your work.

Part 4: Test the business process diagram by using Playback and Inspector

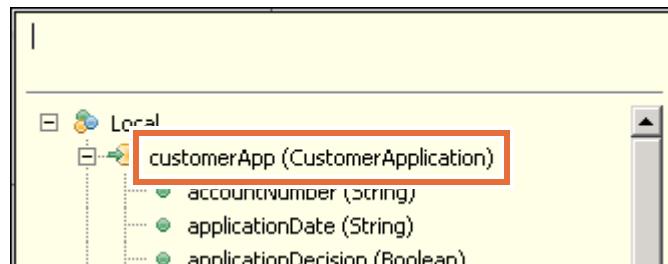
In this part of the exercise, you test the business process diagram in the Inspector view by using Playback tools.

- ___ 1. Before you can test the business process diagram, you must set some test data. For the purposes of this playback, test data is set in a post-assignment step of the start event.
 - ___ a. In the Determine Applicant Eligibility editor, click the **Start** event in the business process diagram to select it.
 - ___ b. In the **Properties** view, click the **Pre & Post** tab. You can set pre-assignments and post-assignments with these options for an activity.
 - ___ c. In the **Pre Assignments** section, click the plus (+) sign to add a pre-assignment step.



- ___ d. On the left side of the equation, click the variable icon.

- __ e. Double-click `customerApp` (`CustomerApplication`) to add it to the equation.



- __ f. On the right side of the equation, type:

```
new tw.object.CustomerApplication()
```

This pre-assignment step initializes the input variable with a new `CustomerApplication` business object.

- __ g. Click the plus sign (+) again to add a second pre-assignment command.
 __ h. On the left side of the equation, click the variable icon.
 __ i. Double-click `customerAppOut` (`CustomerApplication`) to add it to the equation.
 __ j. On the right side of the equation, type:

```
new tw.object.CustomerApplication()
```

This pre-assignment step initializes the output variable.

Pre Assignments

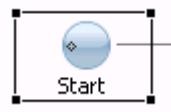
<code>tw.local.customerApp</code>	<code>new tw.object.CustomerApplication()</code>
<code>tw.local.customerAppOut</code>	<code>new tw.object.CustomerApplication()</code>



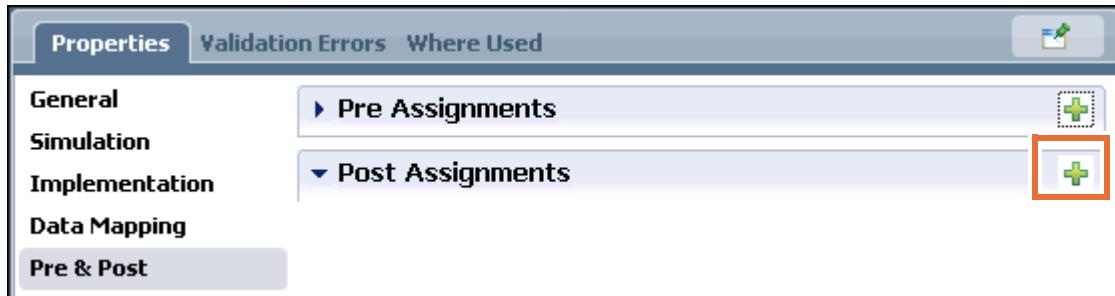
Information

When adding pre-assignment and post-assignment steps to a business process diagram, it is not necessary to add a semicolon (;) at the end of each statement. The IBM Process Designer compiler inserts them for you at compile time.

- __ k. Save your work. A small circular icon is shown on the **Start** event to indicate a pre-assignment option.



- __ I. In the **Post Assignments** section, click the plus (+) sign to add a post-assignment step.



- __ m. On the left side of the equation, click the variable icon.
- __ n. Double-click `companyName (String)` of `customerApp (CustomerApplication)` variable to add it to the equation.
- __ o. On the right side of the equation, type: "AbcCo"

You test the process diagram by using the first path: the ineligible application.

Pre Assignments

<code>tw.local.customerApp</code>	<code>new tw.object.CustomerApplication()</code>
<code>tw.local.customerAppOut</code>	<code>new tw.object.CustomerApplication()</code>

Post Assignments

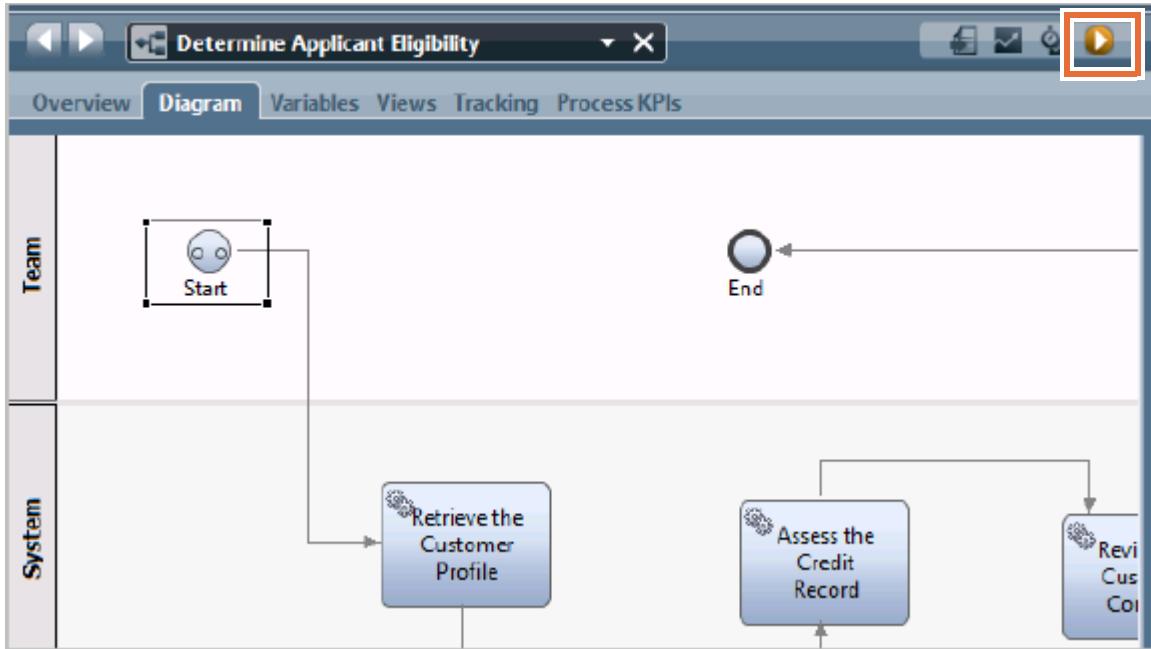
<code>tw.local.customerApp.companyName</code>	<code>"AbcCo"</code>
---	----------------------

- __ p. Save your work.
- __ 2. In the upper-right corner of the process design view, click the **Run Process** icon to test the playback of this process diagram. The Inspector perspective opens to test the playback.

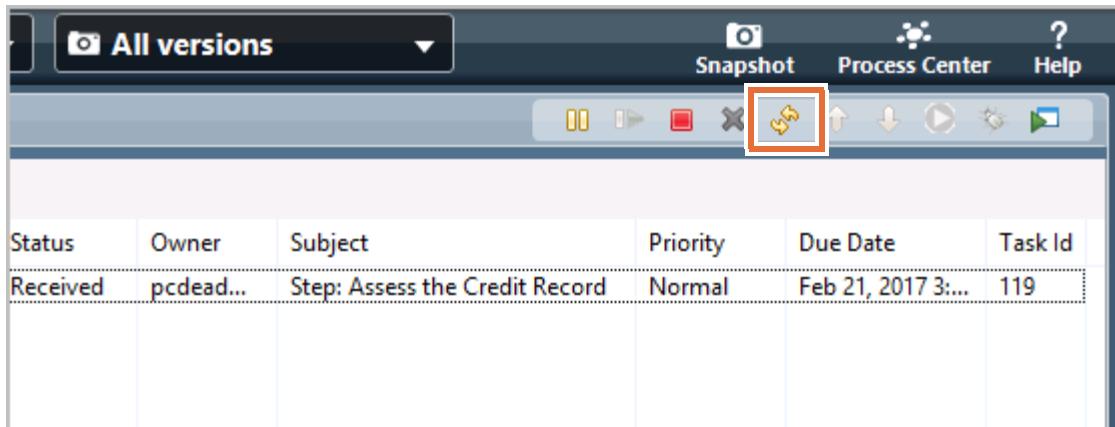


- __ 3. Click **Yes** in the **Switch View** dialog box.

- 4. Click the Run Process icon.

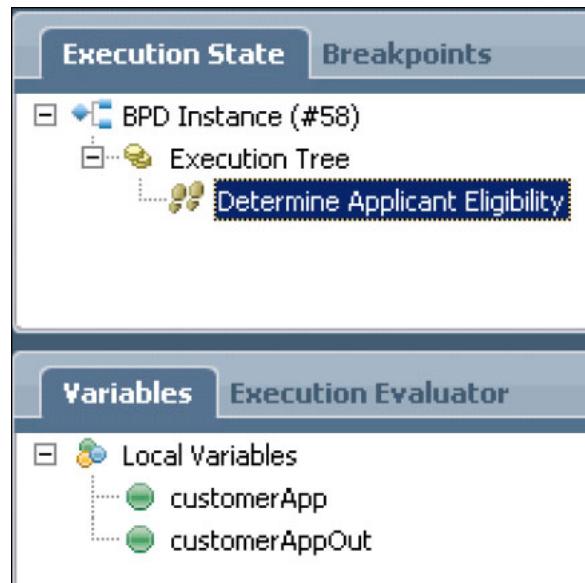


- 5. The process stops at the **Assess the Credit Record** step. Click the refresh icon from the Inspector toolbar.



All the tasks are automated. Inspector moves from the **Retrieve the Customer Profile** activity to the **Assess the Credit Record** activity. It then moves from **Assess the Credit Record** to **Review the Customer Contract**, and from **Review the Customer Contract** to the **End** to complete the process.

- ___ 6. Review the output data to determine whether the values were correctly set.
 - ___ a. In the **Execution State** view, click the **Determine Applicant Eligibility** instance to select it. The variables for the process instance are shown.



- ___ b. Double-click the **customerAppOut** variable. Examine the output values of the variable.

- c. **AbcCo** is considered to be an ineligible customer, so the `eligibleApplication` field must be `false`. Examine the output values.

The screenshot shows the BPD Inspector interface. At the top, the 'Execution State' tab is selected, showing a BPD Instance (#209) and an Execution Tree node 'Determine Applicant Eligibility'. Below this, the 'Variables' tab is selected, showing 'Local Variables' with 'customerAppOut' highlighted by a red box. The 'Execution Evaluator' tab is also visible. A large red box highlights the XML code for the 'customerAppOut' variable:

```
<variable type="CustomerApplication">
<accountNumber type="String"><![CDATA[ABC001]]></accountNumber>
<applicationDate type="String"><![CDATA[2/21/17]]></applicationDate>
<applicationDecision type="Boolean"><![CDATA[false]]></applicationDecision>
<comments type="String"><![CDATA[Bad credit]]></comments>
<companyName type="String"><![CDATA[AbcCo]]></companyName>
<contactFirstName type="String"><![CDATA[Fernando]]></contactFirstName>
<contactLastName type="String"><![CDATA[Torres]]></contactLastName>
<contactPhoneNumber type="String"><![CDATA[312-555-9725]]></contactPhoneNumber>
<creditRating type="String"><![CDATA[F]]></creditRating>
<creditReportNeeded type="Boolean"><![CDATA[true]]></creditReportNeeded>
<creditScore type="Integer"><![CDATA[0]]></creditScore>
<customerCity type="String"><![CDATA[Madrid]]></customerCity>
<customerCountry type="String"><![CDATA[Spain]]></customerCountry>
<eligibleApplication type="Boolean"><![CDATA[false]]></eligibleApplication>
<ineligibleReason type="String"><![CDATA[Bad credit]]></ineligibleReason>
```



Optional

Test the process in the Inspector by using the other acceptable values: **IBM**, **TestCo**, and **ACME**.

- Return to the Designer view.
- Click the **Start** event from the diagram.
- Click the **Pre & Post** option in the **Properties** view.
- Under the **Post Assignments** section, set the right side of the equation to the value you want to test.

The screenshot shows the 'Properties' view with the 'Pre & Post' tab selected. Under the 'Implementation' section, the 'Post Assignments' section is expanded, showing an assignment: `tw.local.customerApp.companyName = "IBM"`. A red box highlights the value "IBM".

- e. Click **Save**.
 - f. Click the **Run Process** icon.
 - g. Follow the preceding steps to complete the test process.

- __ 7. The sample values used for testing must be removed.

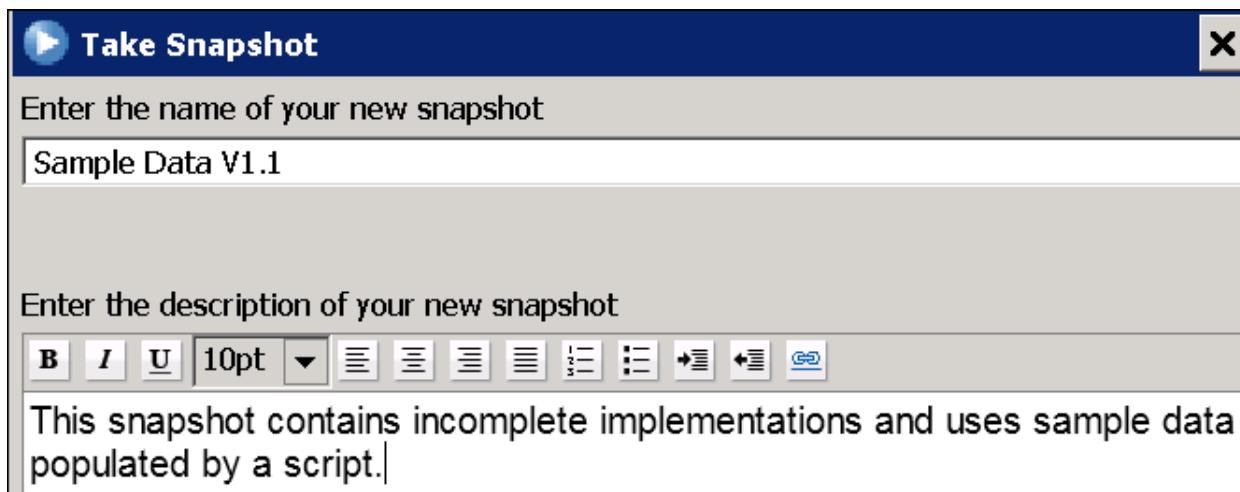
 - __ a. Return to the Designer view.
 - __ b. Click the **Start** event from the business process diagram to select it.
 - __ c. Click the **Pre & Post** option in the **Properties** view.
 - __ d. Under the **Pre Assignments** section, click the **X** to delete the assignment. Repeat this process under the **Post Assignments** section.



- e. Save your work.
 - 8. Take a snapshot of your progress.
 - a. Click the **Snapshot** icon.



- b. Set the name of the snapshot to: Sample Data V1.1
 - c. If you choose to do so, add a description.



- ___ d. Click **OK**. The snapshot is added to the **Revision History**.



- ___ 9. Close IBM Process Designer.

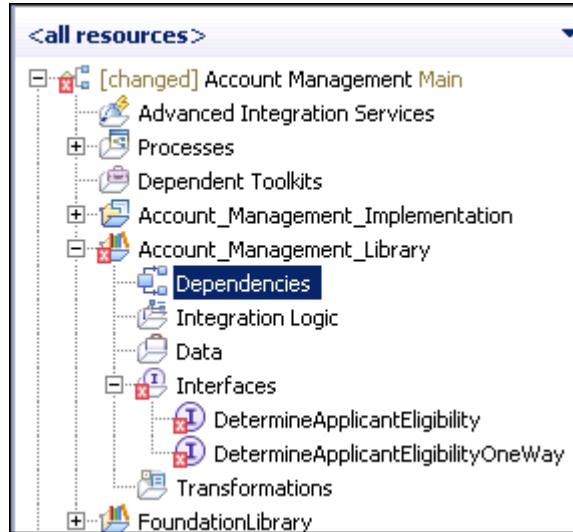
Part 5: Import a business process diagram into IBM Integration Designer

You examined the current functions in the `AccountVerification` scenario. In your exploration of that scenario, you discovered an SCA component with a Java implementation. That component, named `DetermineApplicantEligibility`, was used to set default sample data for a customer application. The sample data is based on the customer's company name. You created a business process diagram in IBM Process Designer, which duplicates this behavior. At the end of the previous section, you stored this model in the IBM Process Center repository with its own snapshot.

In this part of the exercise, you import the business process diagram in to IBM Integration Designer so that it can be used as an SCA import component in your solution.

- ___ 1. If it is not already open, open the Exercise 1 workspace for IBM Integration Designer. Otherwise, skip to the next step.
 - ___ a. On the Windows Desktop, open the folder that is labeled **Exercise Shortcuts**.
 - ___ b. Double-click the shortcut that is named **Exercise 1** to open the workspace.
- ___ 2. Right-click the project that is named `Account Management Main` and click **Refresh from Process Center**. This action updates your working copy with the contents of the IBM Process Center repository. It takes several moments for your working copy to update.
- ___ 3. When a process application is associated, IBM Integration Designer creates an implementation and a library project to use the artifacts of the process application in an assembly diagram. Some of the artifacts that are generated include WSDL interfaces, which are generated in the `Account_Management_Library` subproject. Several errors are associated with these interfaces. Add a dependency on the `FoundationLibrary` to this generated library.
 - ___ a. Expand **Account Management Main > Account_Management_Library**.

- ___ b. Double-click **Dependencies** to open it for editing.



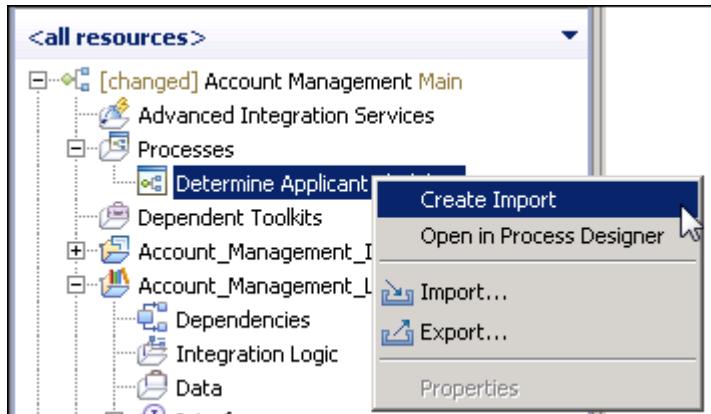
- ___ c. In the **Libraries** section of the dependency editor, click **Add**.
 ___ d. In the **Library Selection** dialog box, select **FoundationLibrary** and click **OK**.
 ___ e. Save your changes. The errors are eliminated.



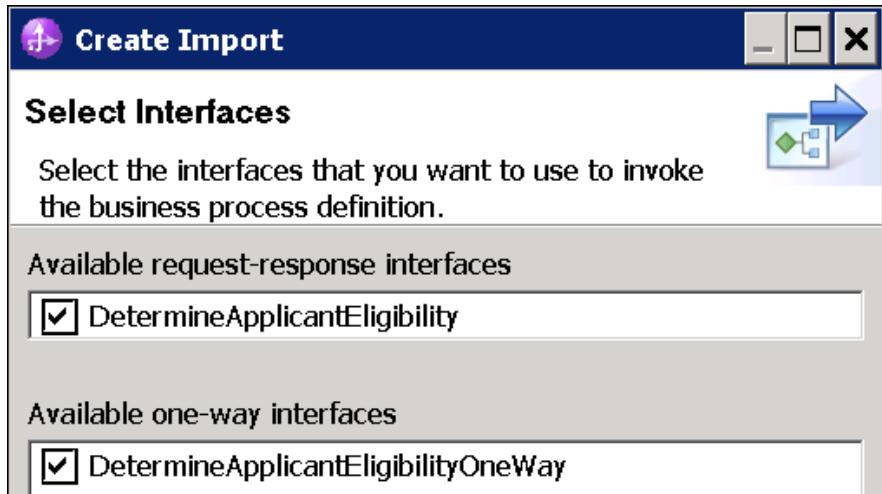
Note

The **FoundationLibrary** is added as a dependency to the **Account_Management_Library** library. Adding a library dependency to an associated process application is not enough. The library must also be mirrored to reflect changes in IBM Process Center and must be shared at the process application level (or global level). You already accomplished the step to mirror the library.

- ___ f. Close the dependency editor.
 ___ 4. After the refresh is complete, expand **Account Management Main > Processes**.
 ___ 5. Right-click the process that is named **Determine Applicant Eligibility** and click **Create Import**.



- 6. In the **Create Import** dialog box, select `DetermineApplicationEligibility` under **Available request-response interfaces** and `DetermineApplicationEligibilityOneWay` under **Available one-way interfaces** and click **Finish**.



An import component is added to the assembly diagram of the `Account_Management_Implementation` subproject.

- 7. Save your work and close the open editor.

Part 6: Republish business process diagram import in IBM Integration Designer

- 1. A [changed] message next to the `Account Management Main` project indicates that the contents of the project were changed. Changes that you made to the project must be published to the IBM Process Center repository. Right-click the project and click **Publish to Process Center**.
- 2. If an XSD resources message is shown, click **OK**. Wait for your changes to finish updating.
- 3. Stop the IBM Process Center cluster environment.
 - a. On the desktop, double-click the shortcut titled **Stop Process Center Cluster**. When the process stops, you are prompted to press any key to continue. Press any key to close the command window.
 - b. On the desktop, double-click the shortcut titled **Stop Process Center node agent**. When the process stops, you are prompted to press any key to continue. Press any key to close the command window.
 - c. On the desktop, double-click the shortcut titled **Stop Process Center deployment manager**. When the process stops, you are prompted to press any key to continue. Press any key to close the command window.
- 4. Close the IBM Integration Designer. Click **File > Exit**, or click **X** in the upper-right corner.

End of exercise

Exercise 2. Version control for SCA applications

Estimated time

01:00

Overview

Version control of modules and libraries is optional in V8.5.7, and if it is used, it is based on a scheme that IBM provides. This exercise demonstrates how to create versioned modules and libraries in IBM Integration Designer for future deployment to a production environment.

Objectives

After completing this exercise, you should be able to:

- Create multiple versions of modules and libraries
- Deploy modules by using the serviceDeploy tool
- Install and start the generated EAR file
- Test the versioned application with cross-component tracing
- Troubleshoot applications in the administration console and Failed Event Manager
- Test the versioned application

Exercise instructions

Introduction

Version control is the ability for the runtime environment to identify snapshots in the lifecycle of a solution or service and to be able to concurrently run multiple snapshots at the same time. Version control is beneficial in a production environment for a number of reasons. Version control allows the simultaneous deployment of multiple versions of an SCA module to the same deployment environment.

You can create versioned modules and libraries in IBM Integration Designer for future deployment to a production environment. These versions are for use on the server at run time, not for distinguishing levels of completion in a development environment. Metadata about module and library version control is stored in the root directory of the project in a file that is called either `sca.module.attributes` or `sca.library.attributes`. The stored data consists of the version number (value), the version provider, and the library dependency.

You can change the versions of modules and libraries for future deployment to a production environment. You can use the default version scheme that IBM Integration Designer provides, a custom scheme, or you can remove the version control completely.

Version control is managed differently within IBM Process Designer. During development, teams use the Snapshot tool to create snapshots, which are stored in the IBM Process Center repository. However, these snapshots are not necessarily deployable. Snapshots that are created in IBM Process Designer might or might not be functional; they capture the process application or toolkit at a particular moment in time.

The IBM Supplied Version Scheme, which is used in IBM Integration Designer, includes major, minor, and service numeric components in the format `<major>.<minor>.<service>` where:

- `<major>` indicates an increment for incompatible changes
- `<minor>` indicates an increment for compatible changes
- `<service>` indicates an increment for bug fixes

An example is 3.1.5.

However, in IBM Process Designer, names, and descriptions of snapshots are arbitrary and do not follow a version control scheme.

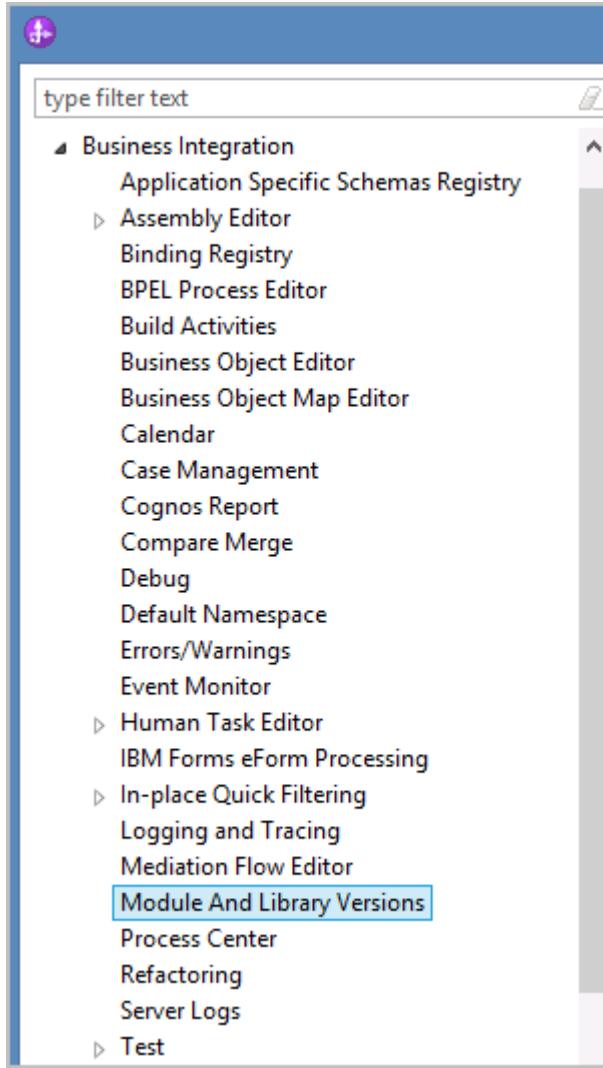
In the production environment, deploying a new version of a module does not replace any previous versions of the module. When a versioned module or library is tested in IBM Integration Designer in the unit test and component test environments, the module is not published to the UTE server as a version-aware module or EAR file. Version control is not applied in the UTE environment.

Part 1: Version the FoundationModule and FoundationLibrary

In this section, you create versions of a module and a library.

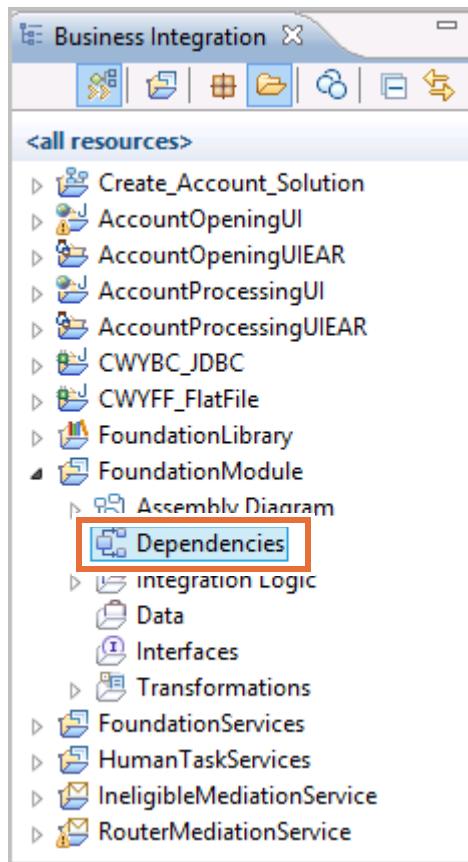
- 1. Open the Exercise 2 workspace.
 - a. On your desktop, open the folder that is labeled **Exercise Shortcuts**.
 - b. Double-click the shortcut that is labeled **Exercise 2**.

- ___ c. When IBM Integration Designer opens, close the **Getting Started** tab.
- ___ 2. View the default version control information in IBM Integration Designer.
 - ___ a. From the menu options, select **Window > Preferences**.
 - ___ b. Expand **Business Integration** and select **Module And Library Versions**.



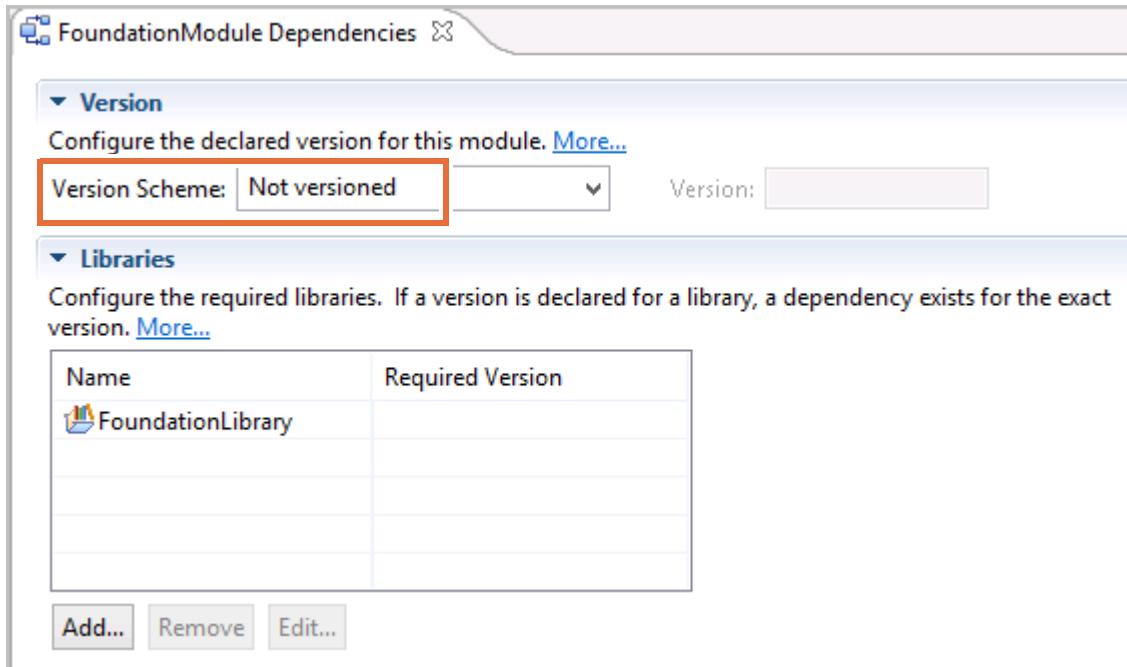
- ___ c. Examine the value in the version scheme field. The default version scheme is **Not versioned**. Making any version scheme modifications in this window does not affect any existing module or library; it applies only to new modules and libraries.
- ___ d. Click **Cancel** to close the preferences window. Since you are working with existing modules and libraries and configure version control manually, you do not have to change the default version control scheme.

- 3. Assign a version number 1.0.1 to **FoundationModule**.
 - a. Expand **FoundationModule** and double-click **Dependencies**.

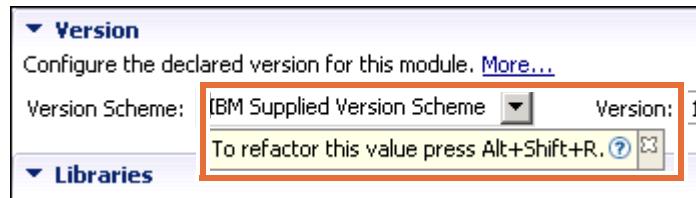


- b. In the **FoundationModule Dependencies** editor, expand the **Version** section at the top to configure the declared version of this module.

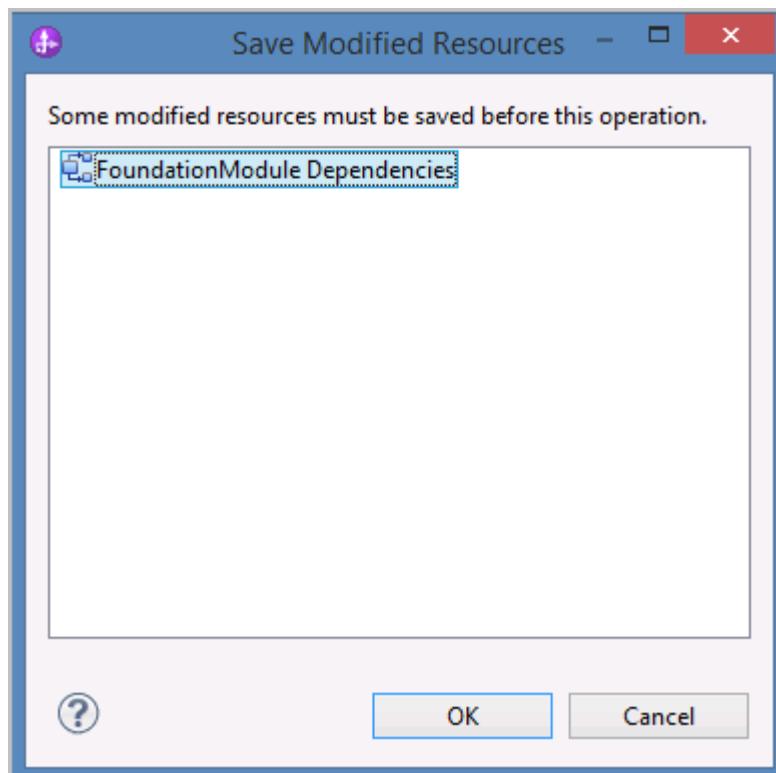
The default **Version Scheme** is **Not Versioned**. When the **IBM Supplied Version Scheme** is selected in the **Preferences** window, newly created modules are assigned a version number automatically, such as 1.0.0.



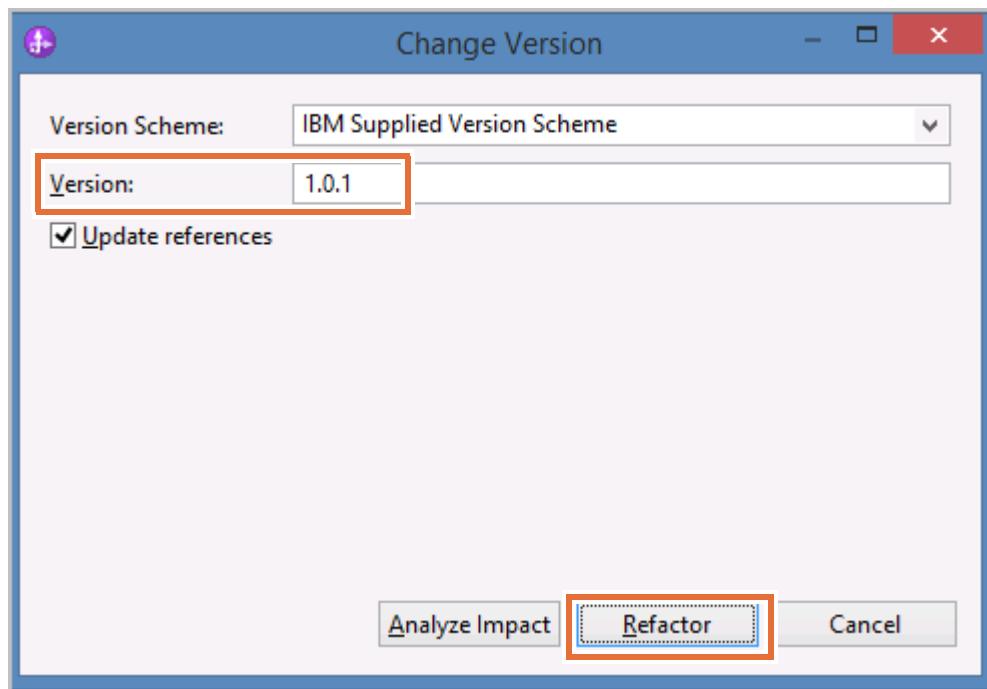
- c. In the **Version Scheme** field, select **IBM Supplied Version Scheme** and press **Alt+Shift+R** to refactor it.



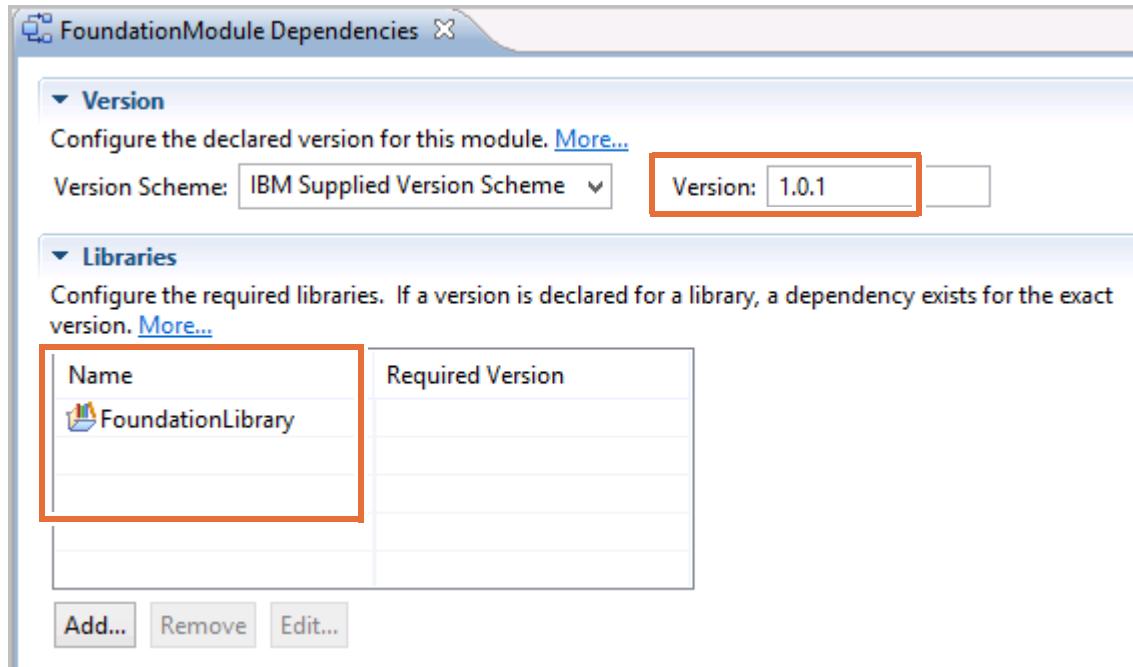
- __ d. In the **Saved Modified Resources** window, select **FoundationModule Dependencies** and click **OK**.



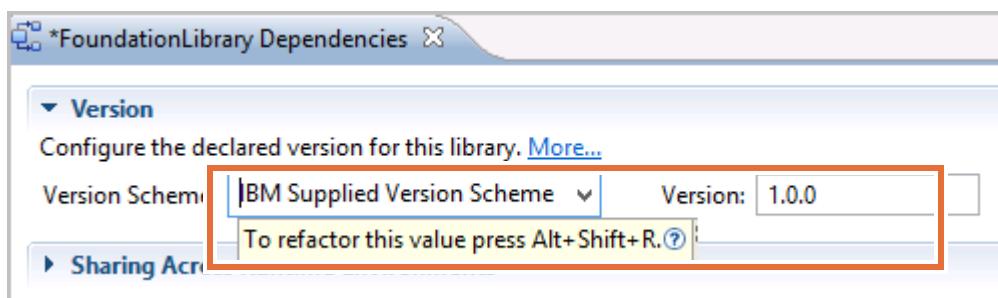
- __ e. Wait for the workspace build to complete. When the build is finished, in the **Change Version** window, enter **1.0.1** in the **Version** field and click **Refactor**.



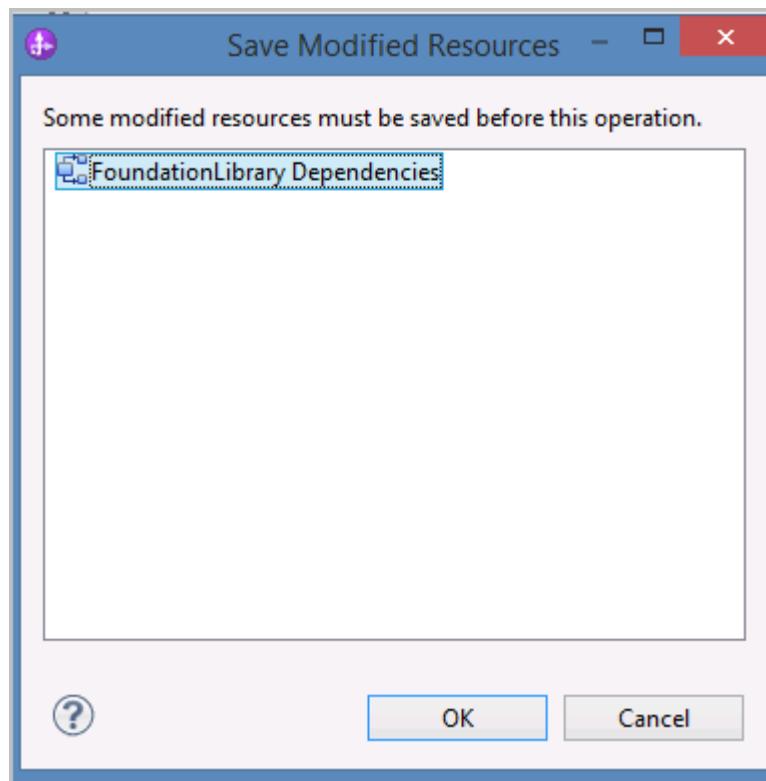
- f. Click **OK** to close the **Concurrent Modules** window. Note the versioned value of the **Foundation Module**, 1.0.1. The **FoundationLibrary** has no **Required Version**. **FoundationModule** is dependent upon **FoundationLibrary**, so it too can have a version number.



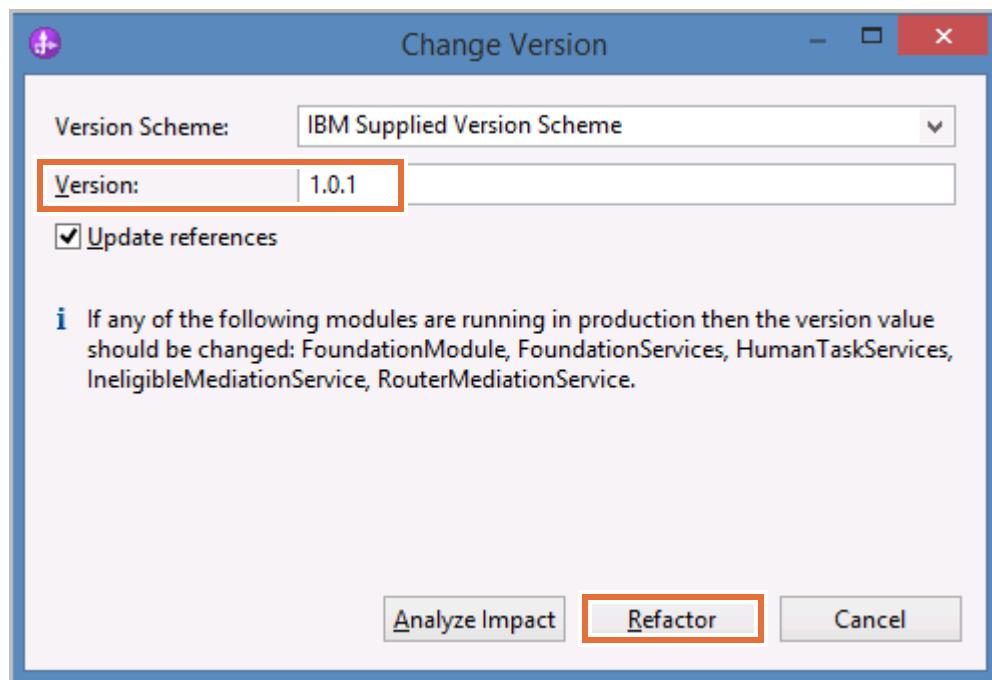
- 4. Assign a version number to the library.
- a. Expand **FoundationLibrary** under Business Integration view and double-click **Dependencies**.
 - b. In the **FoundationLibrary Dependencies** editor, expand the **Version** section.
 - c. In the **Version Scheme** field, select **IBM Supplied Version Scheme** and press **Alt+Shift+R** to refactor it.



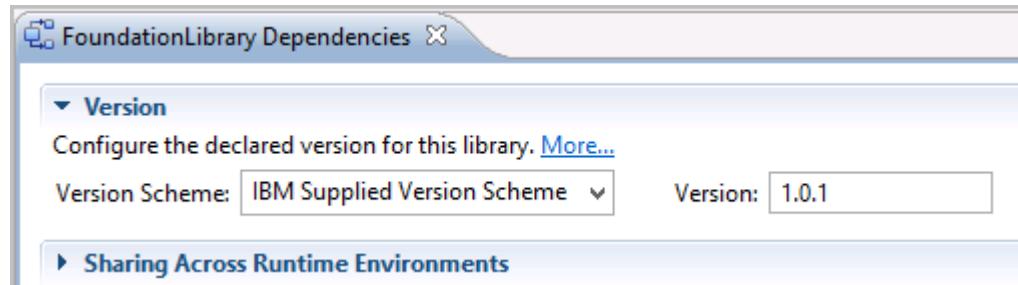
- __ d. In the **Saved Modified Resources** window, select **FoundationLibrary Dependencies** and click **OK**.



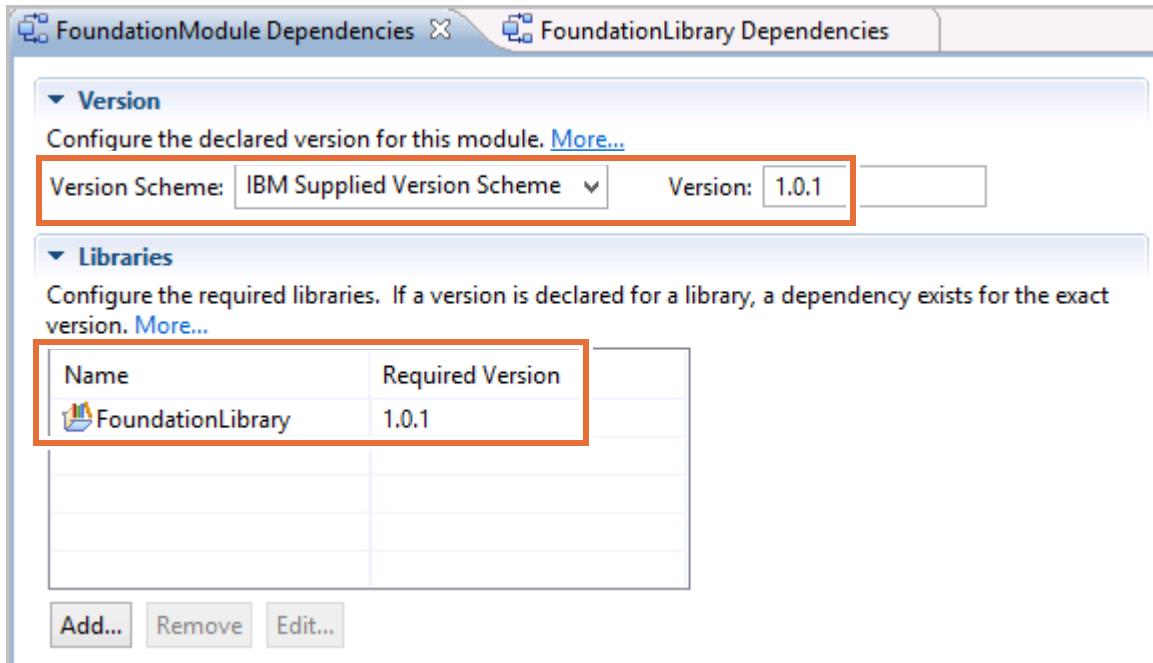
- __ e. Wait for the workspace build to complete. When the build is finished, in the **Change Version** window, change the **Version** to **1.0.1** and click **Refactor**.



- __ f. Examine the versioned value of the **FoundationLibrary**, 1.0.1.

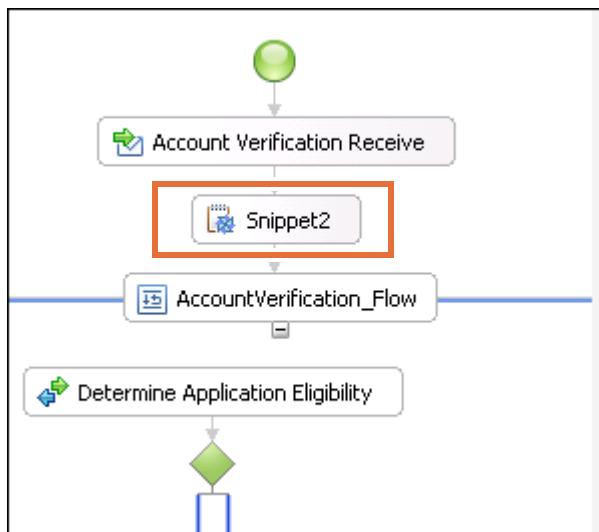


- __ g. Switch to the **FoundationModule Dependencies** tab and note that the **Required Version** of **FoundationLibrary** is updated to 1.0.1.

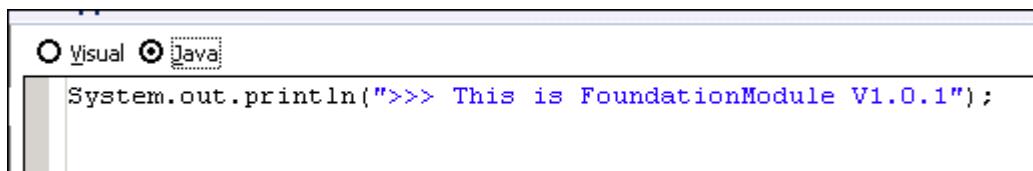


- __ h. Close both dependency tabs.
- __ 5. Add a Java snippet to the **AccountVerification** process that shows the version information in the server console. The console output is used to verify that version 1.0.1 of the application is the one that is running.
- __ a. Expand **FoundationModule > Integration Logic > BPEL Processes > AccountVerification** and double-click **AccountVerification**.

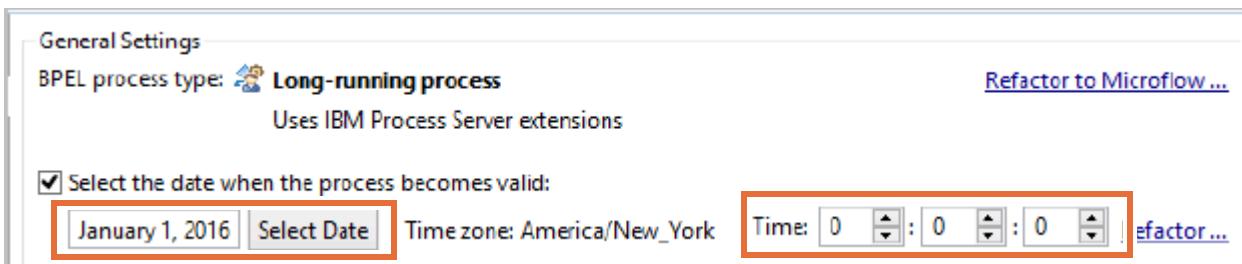
- ___ b. In the palette, expand **Basic Actions** and click **Snippet**. Click below the **Account Verification Receive** to add the **Snippet** activity in the process.



- ___ c. Change the display name of **Snippet2** to: **VersionInfoSnippet**
 ___ d. Switch to the **Properties** view and click the **Details** tab.
 ___ e. Select **Java** (next to the **visual** radio icon).
 ___ f. Click **Yes** in the window to change the snippet type.
 ___ g. Copy and paste the Java code from **C:\labfiles\Support Files\EX2\VersionInfoSnippet.txt** into the snippet window.



- ___ h. Save your work by pressing **Ctrl+S**.
 ___ i. Click an empty area in the BPEL editor canvas.
 ___ j. Switch to the **Properties** view and click the **Details** tab.
 ___ k. Scroll down until you see the **Select the date when the process becomes valid** property. Click the **Select Date** icon and change the date to a date in the past.
 ___ l. Change the time by selecting **0** for **Hours**, **0** for **Minutes**, and **0** for **Seconds**

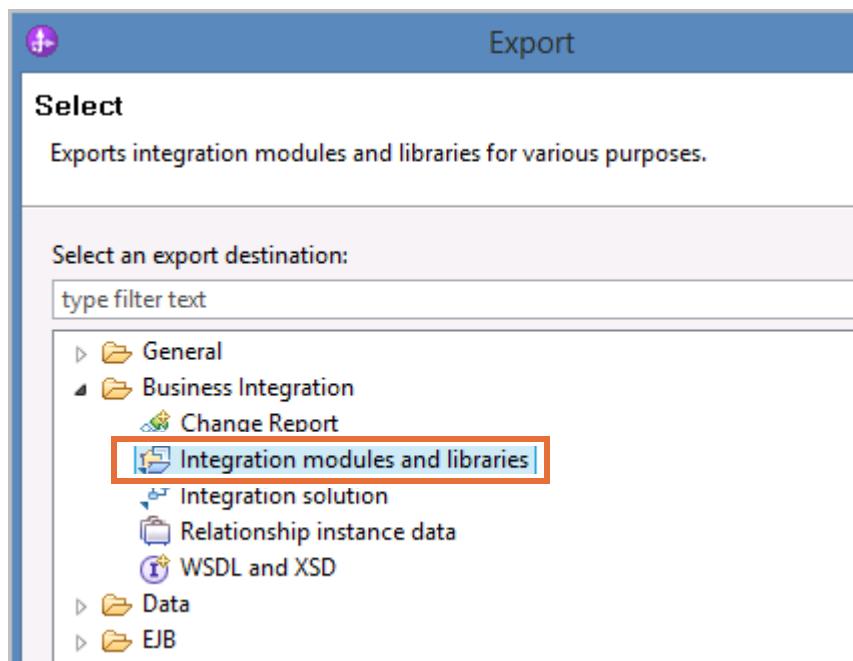


- ___ m. Save your changes and close the open process editor.

Part 2: Deploy FoundationModule with the serviceDeploy tool

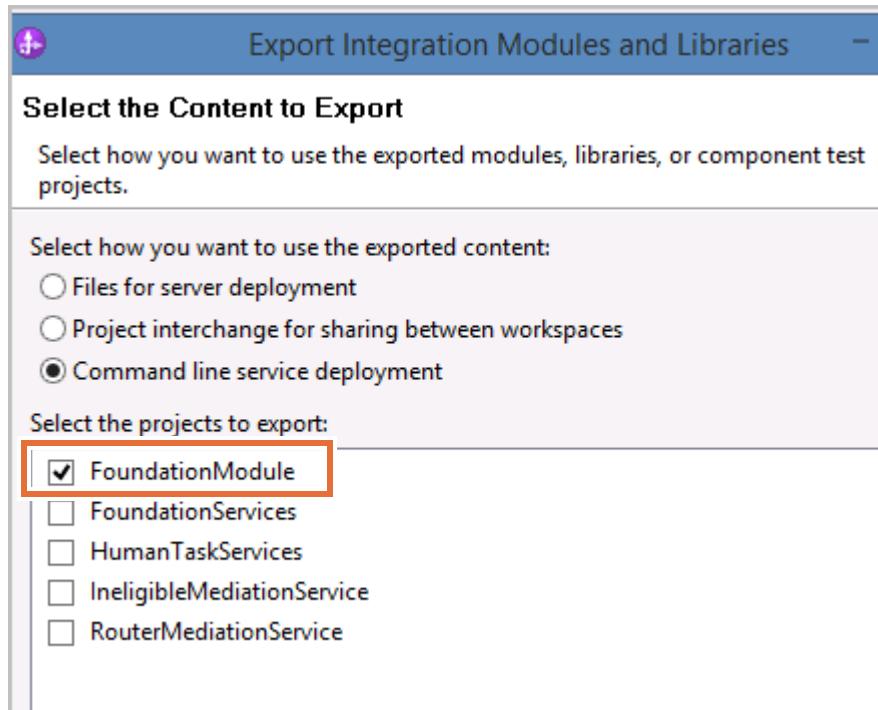
In this section, deploy the module to the server and verify that the version information is included. Versioned modules can be exported for command-line service deployment and deployed with the `serviceDeploy` command to create a version-aware EAR file. Versioned modules that are exported as EAR files are not version-aware. For that reason, the integration module exporter prevents you from exporting versioned modules as EAR files.

- ___ 1. Export **FoundationModule** for command-line service deployment.
 - ___ a. In the Business Integration view, right-click **FoundationModule** and select **Export**.
 - ___ b. In the **Export** window, select **Integration modules and libraries** under **Business Integration**.



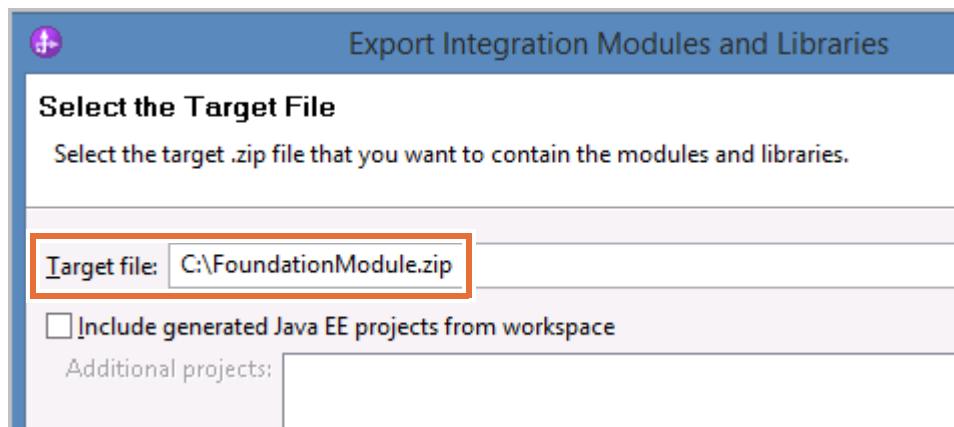
- ___ c. Click **Next**.
- ___ d. Select **Command-line service deployment**.

- __ e. Verify that **FoundationModule** is the only module selected.



- __ f. Click **Next**.

- __ g. In the **Target File** field, enter: C:\FoundationModule.zip



- __ h. Click **Finish**.

- __ 2. Run the `serviceDeploy` command to create a version-aware EAR file.
- __ a. On the desktop, right-click the windows logo in the lower-left corner and click **Run**. Enter `cmd` in the **Run** dialog box, and then click **OK** to start a command window. Optionally, you can click the command prompt shortcut in the taskbar.

- ___ b. Change the working directory by entering `cd C:\IBM\IID\PS\v8.5\bin` and pressing the Enter key.

```
Administrator: C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\Administrator>cd C:\IBM\IID\PS\v8.5\bin
```

- ___ c. Enter the following command, and press Enter:

```
serviceDeploy.bat C:\FoundationModule.zip -outputApplication C:\
```

```
Administrator: C:\Windows\system32\cmd.exe
C:\Users\Administrator>cd C:\IBM\IID\PS\v8.5\bin
C:\IBM\IID\PS\v8.5\bin>serviceDeploy.bat C:\FoundationModule.zip -outputApplication C:\

The workspace is starting in C:\IBM\IID\PS\v8.5\workspace...
The workspace is initializing.
FoundationModule_v1_0_1.zip was successfully imported into the workspace.
The FoundationModule_v1_0_1App project is being created.
The FoundationModule_v1_0_1Web project is being created.
The FoundationModule_v1_0_1 project is building.
FoundationModule_v1_0_1 is being validated.

The following warning message was reported during deployment:

Severity: [warning]
Description: The value of the local variable ret is not used
Resource: com/education/RecordDeclinedApplicationImpl.java
Location: 41

The C:\FoundationModule_v1_0_1App.ear application is being exported.
Deployment has completed.
The workspace is being deleted.

C:\IBM\IID\PS\v8.5\bin>
```

- ___ d. After the command runs successfully, verify that the .ear file is generated under `C:\`.
The file is named: `FoundationModule_v1_0_1App.ear`
- ___ e. Close the command window.

Part 3: Install and start the generated EAR file

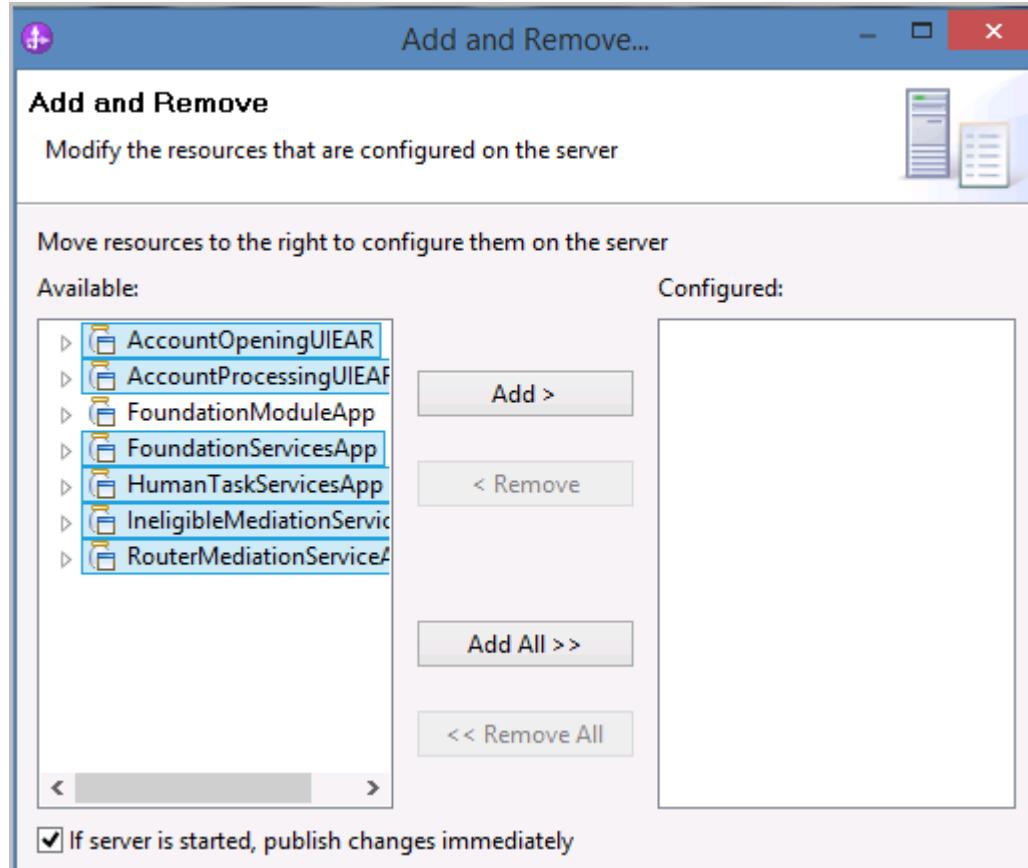
- ___ 1. If IBM Process Server is not running, then start the server by clicking the **Start the server** icon in the **Servers** view. If the **Automatically Publish** window is displayed, click **OK**.



Note

If the server start status shows for more than 5 minutes and the **Server Logs** view is not getting updated with server start messages, then you need to start it manually. Right-click **IBM Process Server v8.5.7 at localhost** and select **Stop** from the menu to stop successfully (do the Stop operation one more time if the server status does not show as Stopped). Restart the server by right-clicking **IBM Process Server v8.5.7 at localhost** and selecting **Start**.

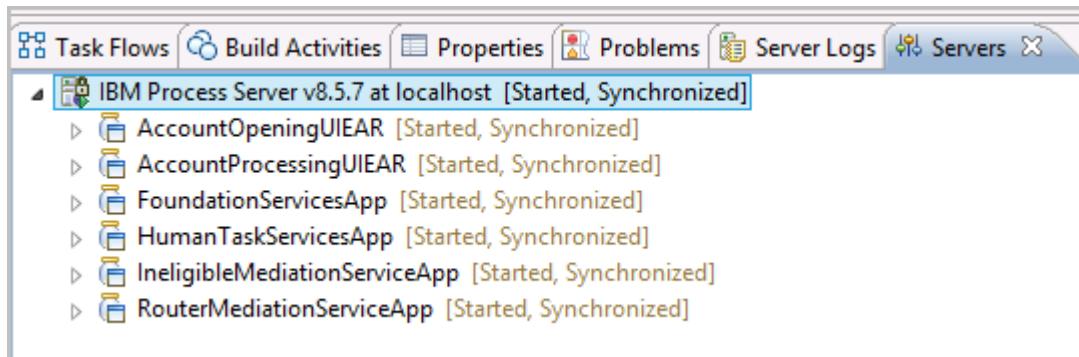
- 2. Install the required applications on IBM Process Server.
 - a. In the **Servers** view, right-click **IBM Process Server v8.5.7 at localhost** and select **Add and Remove** from the menu.
 - b. In the **Add and Remove** window, select all of the projects from the **Available projects selection** area except **FoundationModuleApp**, and click **Add**.



Since you are manually installing the EAR generated for the **FoundationApp** application, you do not deploy it by adding and removing projects.

- c. Click **Finish** to complete the installation of the selected applications. After the installation is complete, the server status and state of the applications change to **[Started, Synchronized]**. The applications on the IBM Process Server are now successfully installed.

- ___ d. In the **Servers** view, expand the + icon next to the server to view the list of applications that were installed.



- ___ e. After verifying that the applications were successfully installed and started, you can collapse the view by clicking the minus (-) icon.



Note

If you see any of the applications not in started state, then in the **Servers** view, right-click **IBM Process Server v8.5.7 at localhost** and select **Publish**.

- ___ 3. Install the generated EAR file on the server.
- ___ a. Start the administration console. In the **Servers** view, right-click the **IBM Process Server v8.5.7 at localhost** server and click **Administration > Run Administrative Console**.



Hint

The administration console is a web-based application. You can start it from a browser by using the following URL:

`https://<hostname>:<port>/ibm/console/logon.jsp`

In the virtual machine environment, this URL is realized as:

`https://localhost:9043/ibm/console/logon.jsp`

- ___ b. Click **Yes** twice, when you see security warnings.
- ___ c. At the **Welcome** page, use `admin` for **User ID** and `web1sphere` for **Password**.
- ___ d. Click **Log In**.
- ___ e. Click **Applications > SCA modules**.
- ___ f. Click **Install**.
- ___ g. Click **Browse** and select `C:\FoundationModule_v1_0_1App.ear`.

- __ h. Click **Open**.
- __ i. Click **Next**.
- __ j. Wait until the installation completes successfully.

CWSCA3017I: Installation task "EJB NamespaceBinding Resource Task for SCAlmportBinding" is running.

CWSCA3017I: Installation task "SIBus Destination Resource Task for SCA SOAP/JMS Invocations" is running.

CWSCA3017I: Installation task "Deployment Task for JaxWsImportBinding and JaxWsExportBinding" is running.

CWSCA3014I: Resources for the SCA application "FoundationModule_v1_0_1App" have been configured successfully.

ADMA5113I: Activation plan created successfully.

ADMA5011I: The cleanup of the temp directory for application FoundationModule_v1_0_1App is complete.

ADMA5013I: Application FoundationModule_v1_0_1App installed successfully.

Application FoundationModule_v1_0_1App installed successfully.

To start the application, first save changes to the master configuration.

Changes have been made to your local configuration. You can:

- [Save directly to the master configuration](#).
- [Review changes before saving or discarding](#).

- __ k. Click **Save**.
- __ l. Examine the version information that is shown in the **Version** column.

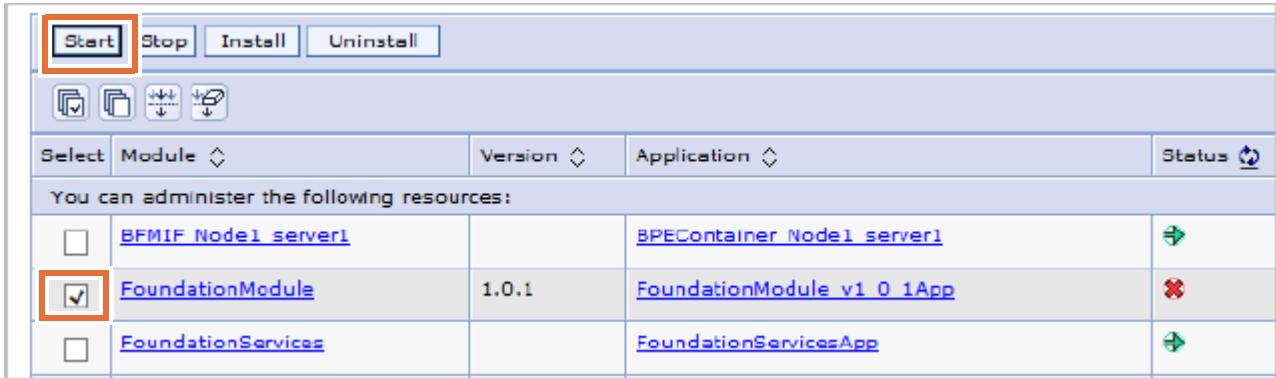
SCA modules

This page shows all installed Service Component Architecture (SCA) modules and their associated services. You can make changes to services without affecting the SCA module services you start the associated application.

+ Preferences

	Start	Stop	Install	Uninstall
Select	Module ◊	Version ◊	Application ◊	
You can administer the following resources:				
<input type="checkbox"/>	BFMIF_Node1_server1		BPEContainer_Node1_server1	
<input type="checkbox"/>	FoundationModule	1.0.1	FoundationModule_v1_0_1App	
<input type="checkbox"/>	FoundationServices		FoundationServicesApp	

- __ m. Select the check box for **FoundationModule** and click **Start**.



The status changes to started, which is represented with a green arrow.

- __ n. Log out of the administration console and close the tab.

Part 4: Test the versioned application with cross-component tracing

About cross-component tracing

By default, the **Server Logs** view shows the standard server console and log records. However, if you enable cross-component tracing, the **Server Logs** view also shows the invocation records that can contain the invocation data that is passed between the components in your application. The invocation records are shown in hierarchical format in the **Server Logs** view, which helps you to more easily understand the relationships that exist between the records. When you enable cross-component tracing, the **Server Logs** view becomes an even more powerful tool for problem determination.

When you enable cross-component tracing on a server, invocation records are generated during Service Component Architecture (SCA) processing of modules and components. The invocation records include information about any errors or events that occurred during processing, such as runtime exceptions. If you choose to enable cross-component tracing with the data snapshot feature, the generated invocation records also contain the invocation input and output data that is passed between the components during processing.

You can enable or disable cross-component tracing for a server from either the **Server Logs** view or the server administrative console. If you enable cross-component tracing from the **Server Logs** view, the tracing is enabled only for the server session. When you next stop or restart the server, the cross-component trace state is automatically disabled by default. By comparison, if you enable cross-component tracing for a server from the server administration console, the cross-component tracing remains enabled for all sessions of the server until you choose to disable it again.

- __ 1. Test the application with the **Create Application** task user interface JSP.
 - __ a. Start Firefox and type the following address in the URL:
`https://localhost:9443/AccountOpeningUI/Index.jsp`
 - __ b. At the login page, use `admin` for the **Name** and `web1sphere` for the **Password**.
This page is a JSP user interface application (the user interface wizard for human tasks generated this page).

- __ c. Click **Login**.



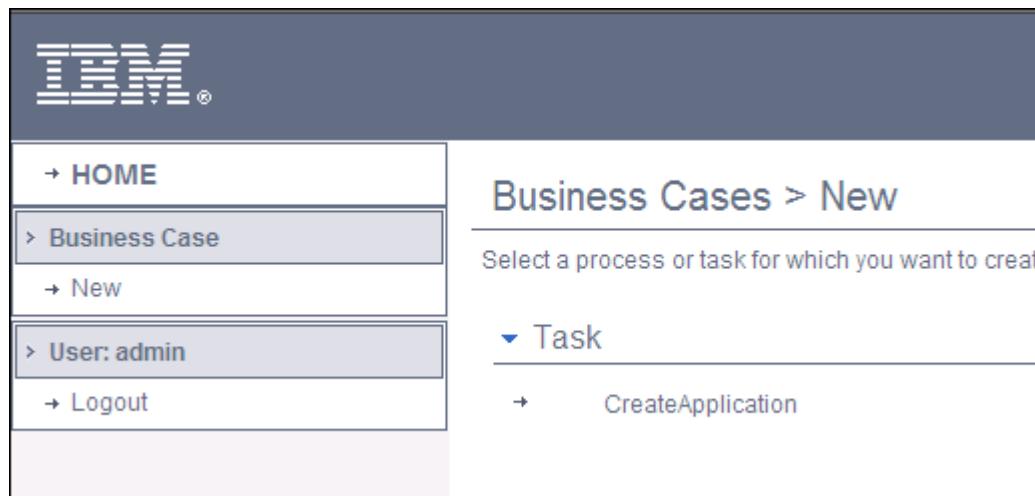
The dialog box has a dark grey header bar with the title "Login to Business User Client". Below it is a light grey content area with the instruction "Enter user name and password, then click Login." In the center, there are two input fields: "Name:" followed by a text box containing "admin", and "Password:" followed by a text box containing "*****". At the bottom is a blue "Login" button.

- __ d. On the **Business User Client** page, click **New**.



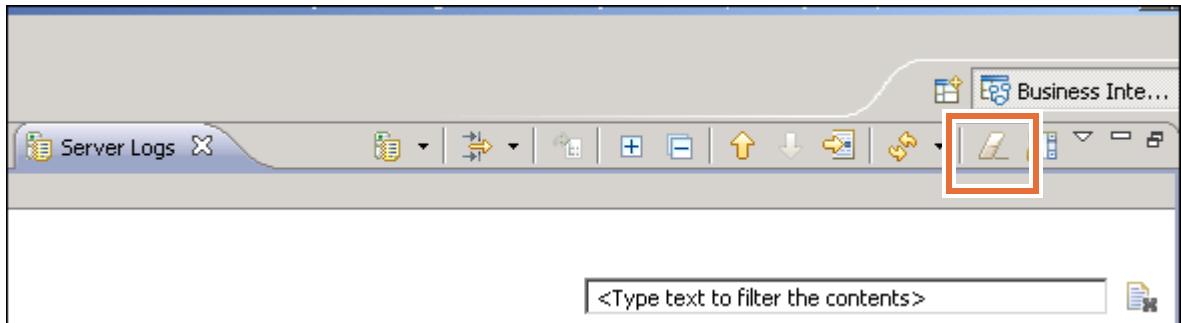
The page features a dark grey header with the IBM logo. A left sidebar contains a navigation menu with items: "HOME" (selected), "Business Case" (selected), "New", "User: admin", and "Logout". The main content area has a title "Business User Client" and a section titled "Business Case" with a "New" link. A note below says "Select this to view a list of all the tasks that you can use to enter information."

- __ e. **CreateApplication** is the only available task. Click the link.



The page has a dark grey header with the IBM logo. The left sidebar includes "HOME", "Business Case" (selected), "New", "User: admin", and "Logout". The main content shows "Business Cases > New" and a note "Select a process or task for which you want to create a new instance". A section titled "Task" contains a single link "CreateApplication".

- f. Switch to the **Server Logs** view in IBM Integration Designer and clear the logs before you start the test. In the menu bar in the **Server Logs** view, click the Clear Server Console icon.



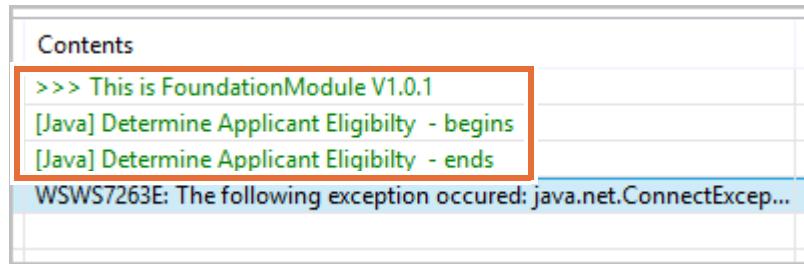
Note

Clearing the **server logs** is an optional task. However, you do not necessarily need to do this step. It makes it easier to find certain output messages during the test.

- g. Switch back to the **CreateApplication** page in the browser and enter **IBM** as the **companyName**. Leave the rest of the fields blank and click **Create** at the bottom of the page.

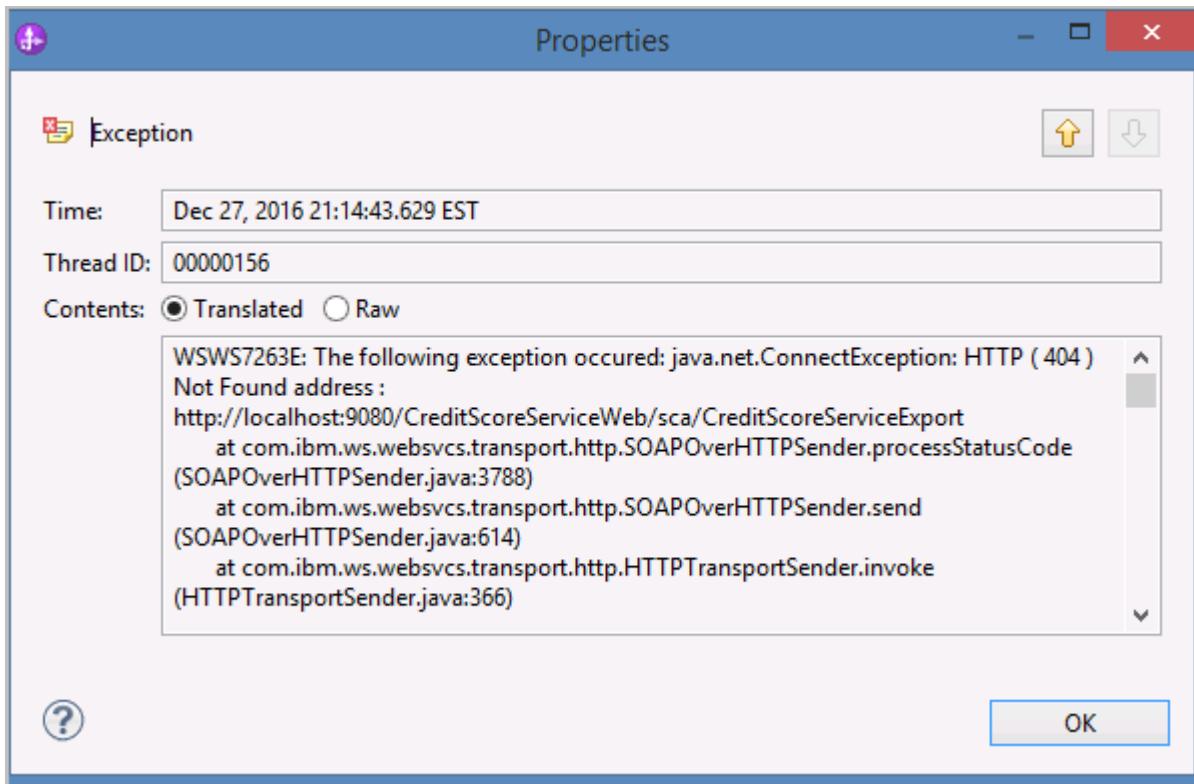
Business Cases > New > CreateApplication	
Enter the values for the input data and optionally provide additional information	
▼ Input Data	
accountNumber	<input type="text"/>
applicationDate	<input type="text"/>
applicationDecision	<input type="checkbox"/>
comments	<input type="text"/>
companyName	<input type="text" value="IBM"/>
contactFirstName	<input type="text"/>
contactLastName	<input type="text"/>
contactPhoneNumber	<input type="text"/>
creditRating	<input type="text"/>

- h. Switch back to the **Server Logs** view. A message is shown which confirms that version 1.0.1 is being called. It is the same text that you added to the **VersionInfoSnippet** in your business process.



The first test that verifies version information is complete. You next create a version of the application and test it.

- i. Double-click the last row in the **Server Logs** view to view its properties. It indicates a problem with the request (an Exception).
 — j. In the Properties window view, the error message. It shows the runtime error that the CreditScoreServiceExport returned.

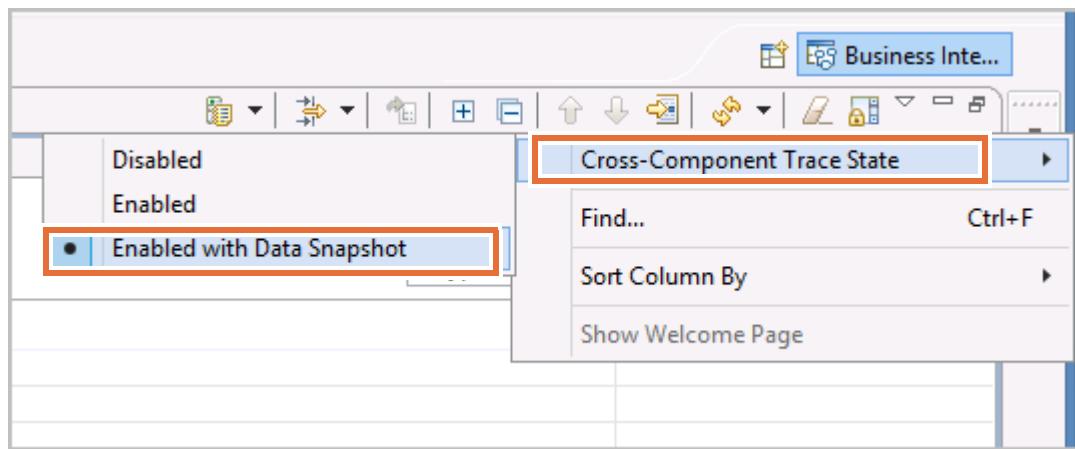


- k. Click **OK** to close the message.

- 2. Enable cross component trace before you test again. You can view errors and exception details that help you resolve any problem.
- a. Access cross component trace settings by clicking the **View menu** at the far-right upper corner of the **Server Logs** view.



- b. Enable cross component trace by selecting **Cross-Component Trace State > Enabled with Data Snapshot** from the view menu.



Information

The **Enabled with Data Snapshot** option allows cross-component tracing with the data snapshot feature. Invocation records are generated into both the server console and the `System.out` and `trace.log` files, but the record properties do include invocation input and output data. This data is captured in input and output files under the `logs\xct` directory.

The **Enabled** option allows cross-component tracing. Invocation records are generated into both the server console and the `SystemOut.log` and `trace.log` files, but the record properties do not include any invocation input and output data. The `SystemOut.log` and `trace.log` files are in the server log directory.

The current cross-component trace state is always shown in the lower-right corner of the **Server Logs** view. You can enable or disable cross-component tracing for a server from either the **Server Logs** view or the server administrative console. If you enable cross-component tracing from the **Server Logs** view, the tracing is enabled only for the server session. When you next stop or restart the server, the cross-component trace state is automatically disabled again by default. Another option is to enable cross-component tracing for a server from the server administration console. With this method, the cross-component tracing remains enabled for all sessions of the server until you choose to disable it again.

**Note**

You might want to clear the server logs again before running the test to have a better view.

- 3. Run the test again.
 - a. From the JSP page, send the test data for company IBM again. Now that you enabled cross component trace, you can see the event flow.
 - b. In the **Servers Logs** view, expand the server output if it is collapsed by clicking the + icon as shown in the diagram.

Show All Record Types (Hierarchical) > with only Server State and Error Contents (Page 1 of 1)		
Type	Thread ID	Contents
Invocation sequence (AccountVerification:InputCriterion)	00000112	

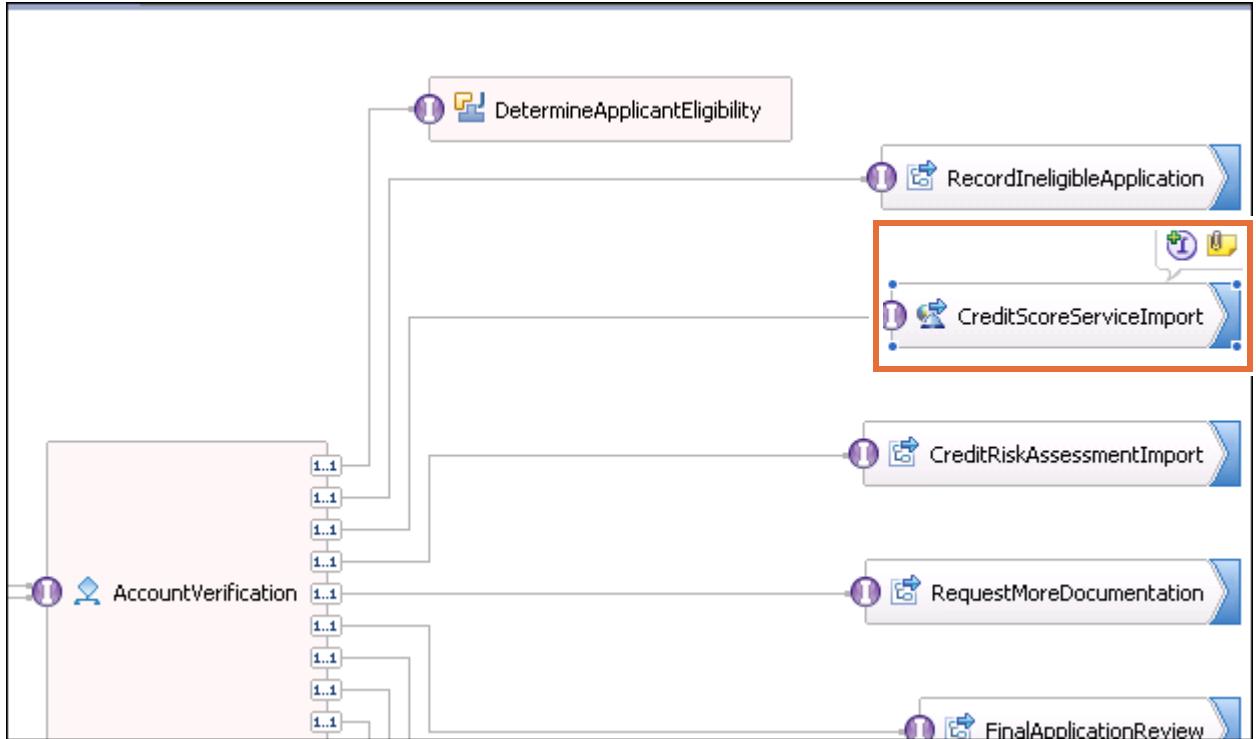
- c. Drill down in the output until you see the **Exception** message.

Console (filtered): IBM Process Server v8.5.7 at localhost			
Show All Record Types (Hierarchical) > with only Server State and Error Contents (Page 1 of 1)			
Type	▲	T...	Contents
Start HTM (_TKI:a01b0159.433fade0.a8afe53.d61)	De...	00...	Start of processing for HTM (_TKI:a01b0159.433fade0.a8afe53.d61)
Start invoke (AccountVerification:InputCriterion)	De...	00...	Start of the asynchronous invocation of operation AccountVerific...
Start component (AccountVerification:ln)	De...	00...	Start of the component processing of operation AccountVerific...
In BPEL process	De...	00...	07e8a3dd-ddab-4640-be64-54345fc78d94 STATE 174e8ac8-6...
Start BPEL process (AccountVerification:Pl)	De...	00...	Start of processing for BPEL process AccountVerification:_Pl:9...
End BPEL process (AccountVerification:Pl)	De...	00...	End of processing for BPEL process AccountVerification:_Pl:9...
Start BPEL process (AccountVerification:Pl)	De...	00...	Start of processing for BPEL process AccountVerification:_Pl:9...
End BPEL process (AccountVerification:Pl)	De...	00...	End of processing for BPEL process AccountVerification:_Pl:9...
Start BPEL process (AccountVerification:Pl)	De...	00...	Start of processing for BPEL process AccountVerification:_Pl:9...
Start invoke (CreditScoreServiceImport)	De...	00...	Start of the invocation of operation CreditScoreServiceImport...
Start import (CreditScoreServiceImport)	De...	00...	Start of the import processing of operation CreditScoreService...
Exception	De...	00...	WSWS7263E: The following exception occurred: java.net.Con...
Fail import (CreditScoreServiceImport)	De...	00...	A failure occurred during the import processing of operatio...
Fail invoke (CreditScoreServiceImport)	De...	00...	A failure occurred during the invocation of operation Credit...
End BPEL process (AccountVerification:Pl)	De...	00...	End of processing for BPEL process AccountVerification:_Pl:9...
End component (AccountVerification:ln)	De...	00...	End of the component processing of operation AccountVerific...
End invoke (AccountVerification:InputCriterion)	De...	00...	End of the asynchronous invocation of operation AccountVerifi...
End HTM (_TKI:a01b0159.433fade0.a8afe53.d61)	De...	00...	End of processing for HTM (_TKI:a01b0159.433fade0.a8afe53...

- d. You see two fail messages, one below the other. One is **Fail import** and the other is **Fail invoke**.

Start BPEL process (AccountVerification)
Start invoke (CreditScoreServiceImport:calculateCreditScore)
Start import (CreditScoreServiceImport:calculateCreditScore)
Exception
Fail import (CreditScoreServiceImport:calculateCreditScore)
Fail invoke (CreditScoreServiceImport:calculateCreditScore)
End BPEL process (AccountVerification)

- __ e. Right-click **Fail Import (CreditScoreServiceImport:calculateCreditScore)** and select **Show Component in Assembly Diagram** from its menu.
 - __ f. The **FoundationModule** Assembly diagram opens up and the **CreditScoreServiceImport** component is selected. Click the **Properties** tab and by default the properties of this component are available to view.



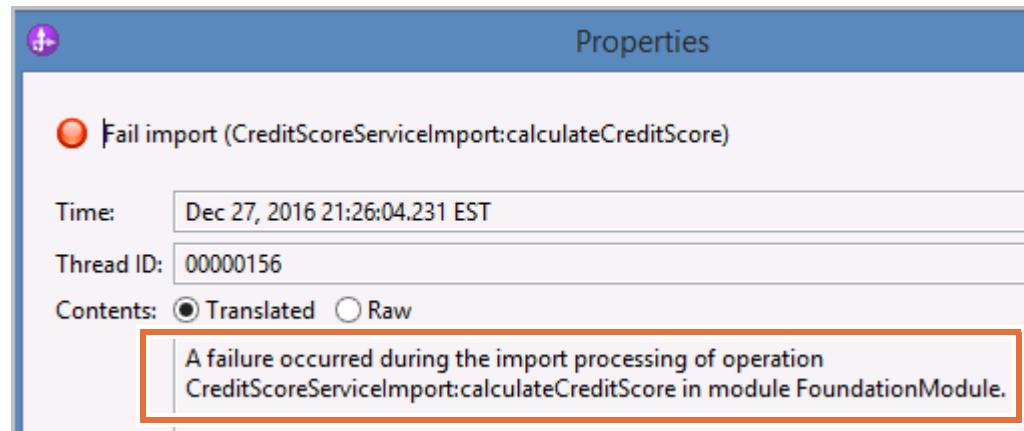
According to the log entry, the point of failure occurs here.

- g. Click **Binding** to review bindings information for this component.

Import: CreditScoreServiceImport (Web Service Binding)

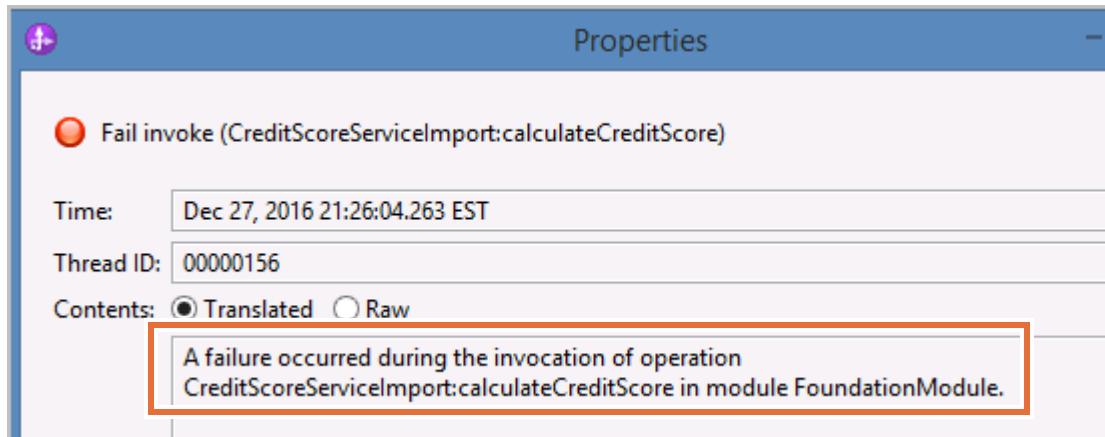
Description	Transport:	SOAP1.2/HTTP
Details	Address:	<code>http://localhost:9080/CreditScoreServiceWeb/sca/CreditScoreServiceExport</code>
Binding	Port:	CreditScoreServiceExport_CreditScoreServiceHttpPort
JAX-WS Handlers	Service:	CreditScoreServiceExport1_CreditScoreServiceHttpService

- h. Return to the **Server Logs** view and double-click the **Fail Import** entry to view its properties.



The message indicates, “A failure occurred during the import processing of operation CreditScoreServiceImport:calculateCreditScore in module FoundationModule.”

- i. Click the down-arrow key to view the next message. The **Fail Invoke (CreditScoreServiceImport:calculateCreditScore)** shows the error message.



The content includes the following message:

A failure occurred during the invocation of operation
CreditScoreServiceImport:calculateCreditScore in module FoundationModule.



Note

The message indicates that the import component failed to connect to the web service, **CreditScoreServiceExport**. Therefore, the next step is to verify the accuracy of the web service address, and the availability of the web service.

- j. Click **OK** to close the **Properties** window.

Part 5: Troubleshoot applications in the administration console and Failed Event Manager

- ___ 1. Start the administrative console to check whether the CreditReport service exists.
- ___ a. Start Firefox and enter the following address to open the administrative console:
<https://localhost:9043/ibm/console>



Hint

You can also start the administrative console by right-clicking the server in the **Servers** view and clicking **Administration > Run administrative console**.

- ___ b. If you receive the **This Connection is Untrusted** page, click **I Understand the Risks**, then click **Add Exception**. In the **Add Security Exception** window, click **Confirm Security Exception**.
- ___ c. At the login page, use **User ID** `admin` and **Password** `web1sphere` and click **Log in**.
- ___ d. Select **Applications > SCA modules**.
- ___ e. Review the list of applications that are running on the server. No `CreditCheckService` SCA application is on the server.

You can administer the following resources:			
<input type="checkbox"/>	BFMIF_Node1_server1		BPEContainer_Node1_server1
<input type="checkbox"/>	FoundationModule	1.0.1	FoundationModule_v1_0_1App
<input type="checkbox"/>	FoundationServices		FoundationServicesApp
<input type="checkbox"/>	HTMIF_Node1_server1		TaskContainer_Node1_server1
<input type="checkbox"/>	HTM_PredefinedTaskMsg_V8000 (Node1_server1)		HTM_PredefinedTaskMsg_V8000_Node1_server1
<input type="checkbox"/>	HTM_PredefinedTasks_V8000 (Node1_server1)		HTM_PredefinedTasks_V8000_Node1_server1
<input type="checkbox"/>	HumanTaskServices		HumanTaskServicesApp
<input type="checkbox"/>	IneligibleMediationService		IneligibleMediationServiceApp
<input type="checkbox"/>	RouterMediationService		RouterMediationServiceApp
Total 9			

__ 2. Check the entries in the **Failed Event Manager**.

- __ a. In the administration console, select **Servers > Deployment Environments > ProcessServer > Failed Event Manager**.

Deployment Environments

Deployment Environments > ProcessServer

A deployment environment manages a set of resources as defined by its deployment topology pattern.

Configuration

General Properties

- Deployment Environment: ProcessServer
- Deployment Environment Pattern:
- Description:

Additional Properties

- Deferred Configuration
- Failed Event Manager** (highlighted with a red box)
- Health Center
- Process Server Settings

Related Items

- Authentication Aliases

- __ b. Select **Get all failed events** under the **Failed events on this server** section.

Deployment Environments

Deployment Environments > Failed Event Manager

The failed event manager is used to query and manage failed events.

Failed events on this server

The following are some common methods for managing failed events:

- Get all failed events** (highlighted with a red box)
- Search failed events

About your failed event manager

- The Recovery sub-system is enabled.
- Total failed events: 2
- IBM WebSphere Application Server Network Deployment, 8.5.5.8
Build Number: cf081545.03
Build Date: 11/12/15
- Licensed Material - Property of IBM
5724-J08, 5724-I63, 5724-H88, 5724-H89,

- c. Depending on the number of times the application was run and the event that was fired, you see one or more **Event IDs** listed. Click an event ID to view the details.

The screenshot shows a search results page for failed events. The table has columns for Selected, Event ID, Event type, Module, Component, and Operation. Two rows are selected, and their Event IDs ([PI:90030159.433..](#) and [PI:90030159.433..](#)) are highlighted with a red box.

Selected	Event ID	Event type	Module	Component	Operation
<input type="checkbox"/>	PI:90030159.433..	BPC	FoundationModule..	AccountVerification	
<input type="checkbox"/>	PI:90030159.433..	BPC	FoundationModule..	AccountVerification	
Total: 2					

- d. Click any of the hyperlinks. Each hyperlink is associated with a failed process instance. The event type for each of these instances is **BPC**.

The screenshot shows the details of a failed event. The event ID is [PI:90030159.43354700.a8afe53.d61c0091](#). The event type is BPC, which is highlighted with a red box. Other properties shown include event status (stopped), module (FoundationModule_v1_0_1), and component (AccountVerification).

Event ID	PI:90030159.43354700.a8afe53.d61c0091
Event type	BPC
Event status	stopped
Module	FoundationModule_v1_0_1
Component	AccountVerification

- __ e. Click **View Business Data** to view the sent message.
- __ f. Click **Input** to view the parameter values.

Deployment Environments

[Deployment Environments](#) > [Failed Event Manager](#) > [Search results](#) > [PI:90030159.43354700.a8afe53.d61c0091](#) > [Business data editor](#)

Use this page to view and edit business data parameters.

Parameter name	Parameter value	Parameter type
Input	...	CustomerApplication
Total 1		

- __ g. Review the parameter values for this input.

Deployment Environments

[Deployment Environments](#) > [Failed Event Manager](#) > [Search results](#) > [PI:90030159.43354700.a8afe53.d61c0091](#) > [Business data editor](#)

Use this page to view and edit business data parameters.

Parameter name	Parameter value	Parameter type
accountNumber		String
creditReportNeeded	false	Boolean
creditRisk		String
creditScore	0	Integer
customerCity		String
customerCountry		String
eligibleApplication	false	Boolean

- __ h. Click the event ID to go back to the details view. The event ID in your lab is different from the value that is listed in the diagram, but is ordinarily prefixed with the code `_PI:` to indicate a process instance.

Deployment Environments

[Deployment Environments](#) > [Failed Event Manager](#) > [Search results](#) > [PI:90030159.43354700.a8afe53.d61c0091](#) > [Business data editor](#)

Use this page to view and edit business data parameters.

Parameter name	Parameter value	Parameter type
Input	...	CustomerApplication
Total 1		

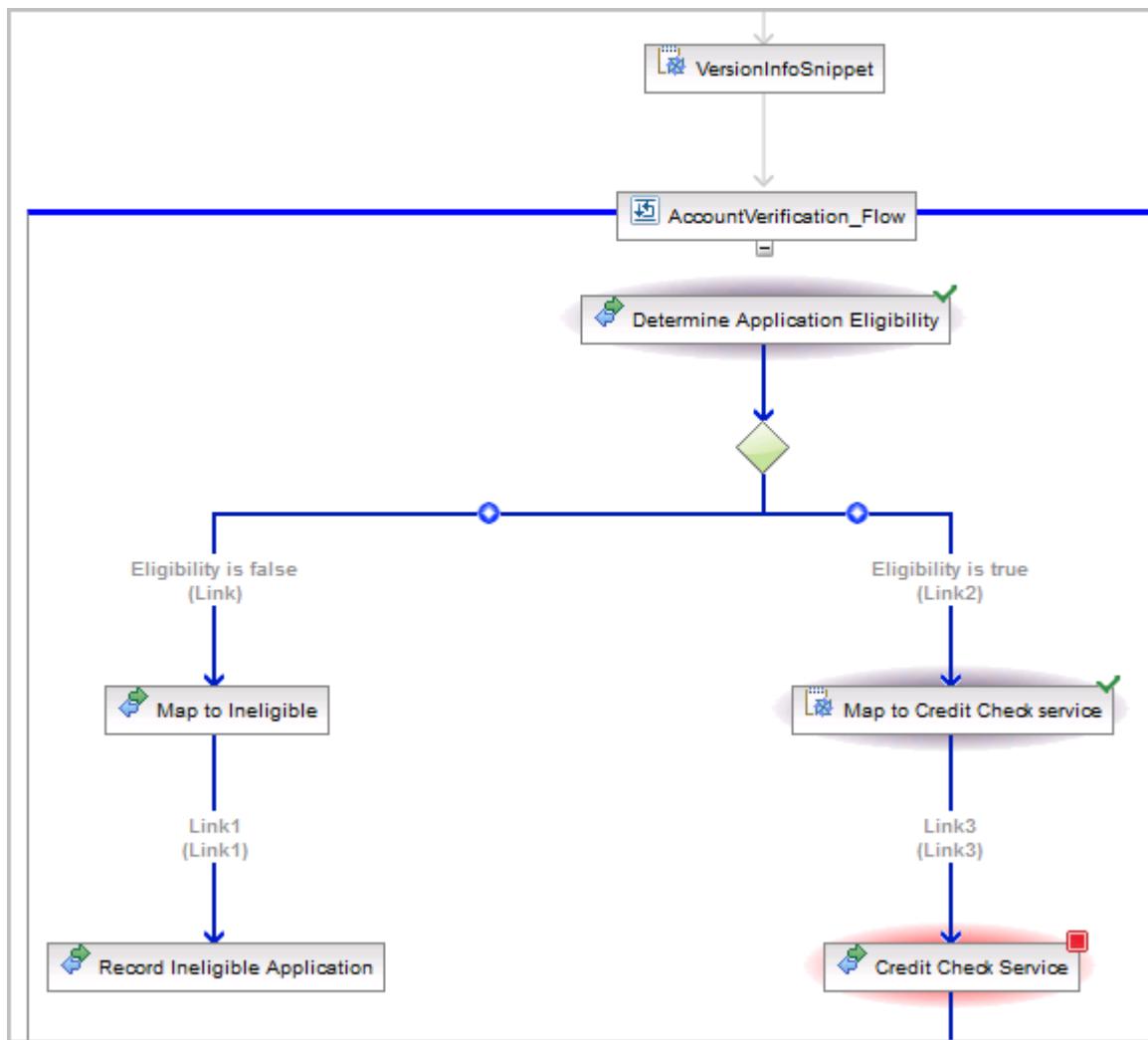
- __ i. Click **Open calling process in Business Process Choreographer Explorer**. The BPEL explorer shows the detailed information about the failed process instance.

The screenshot shows a web-based interface for managing deployment environments. At the top, it says "Deployment Environments". Below that, a breadcrumb navigation path is shown: "Deployment Environments > Failed Event Manager > Search results > _PI:90030159.43354700.a8afe53.d61c0091". A message below the path says "Use this page to view details about the failed event." There is a button labeled "Failed event details". Below that are three buttons: "View business data", "Resubmit", and "Delete". A prominent blue link "Open calling process in Business Process Choreographer Explorer" is highlighted with a red border. Below this, there is a section titled "Failed event common properties" with a sub-section for "Event ID" containing the value "PI:90030159.43354700.a8afe53.d61c0091".

- __ j. If a message is displayed that the connection is not secure, then add the exception to continue.
 __ k. At the login page, use **User Name** admin and **Password** web1sphere and click **Login**.
 __ l. Click **View Process State**.

The screenshot shows a "Process Instance" page. At the top, it says "Process Instance". Below that, a message says "Use this page to view information about a process instance and, optionally, to work on th...". There are five buttons at the top: "Terminate", "Suspend", "Work Items", "Create Work Items", and "View Process State", with the last one highlighted by a red box. Below these buttons is a section titled "Process Description". It contains a table with three rows:
 - Process Instance Name: _PI:90030159.43354700.a8afe53.d61c0091
 - Description: Account verification for IBM
 - State: Running

- __ m. Review the visual display of the process, and note the activities that were successfully run and the activity that failed.

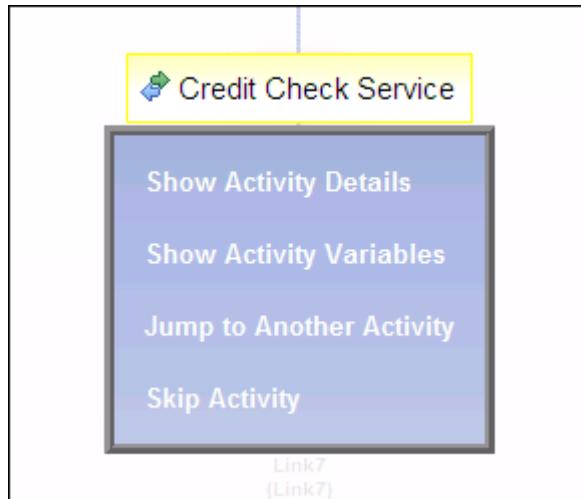


Hint

In the lower-right corner of the process state viewer, a map indicates which part of the process you are viewing.



- __ n. Click **Credit Check Service**. A menu option is shown.



- __ o. Click **Show Activity Details** to view details on the **Credit Check Service**.
- __ p. The **Invoke3** activity is stopped because of failed implementation.

The screenshot shows a web-based interface titled "Activity". The page has a header with the sub-instruction: "Use this page to view details about an activity and its associated process." Below the header is a row of buttons: "Restart", "Force Complete", "Skip", "Jump", and "Variables". Underneath these buttons is a section titled "Activity Description". A table displays activity details:

Activity Name	Invoke3
Kind	Invoke
State	Stopped
Stop Reason	Implementation failed

The "Stop Reason" row is highlighted with a red border. At the bottom of the table, there is a link labeled "Description".

- __ q. Click the **Error Details** tab to view the error details. The error complains that the address is not found.

The screenshot shows the 'Activity' page with the following details:

- Activity Description:**
 - Activity Name: Invoke3
 - Kind: Invoke
 - State: Stopped
 - Stop Reason: Implementation failed
 - Description: (empty)
- Activity Instance ID:** _AI:9004015a.661d6cdb.a8afef53.ec2200c3
- Tab Bar:** Details, Activity Input Message, Activity Output Message, Tasks, Custom Properties, **Error Details** (highlighted with a red box).

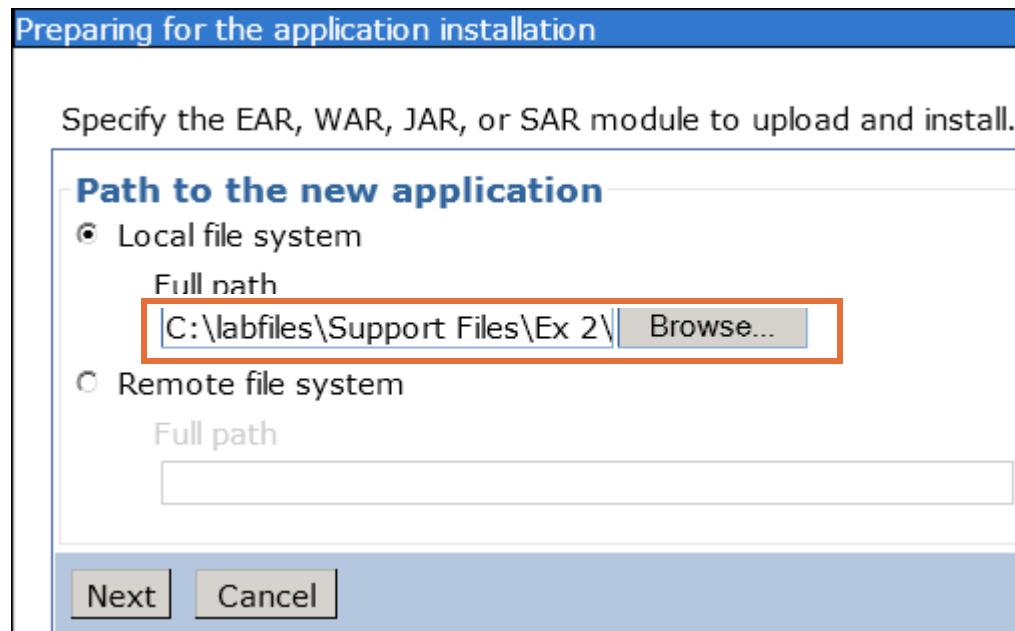
- __ 3. Deploy the missing web service to the server.
- Return to the IBM Process Server administration console.
 - In the administration console, select **Applications > New Application**.
 - Click **New Enterprise Application**.

The screenshot shows the 'New Application' dialog with the following interface:

- Left Sidebar:** View: All tasks, Welcome, Guided Activities, Servers, **New Application** (highlighted with a red box), SCA modules, Services, Resources, Security, Environment, Integration Applications.
- Right Panel:**
 - Cell=qcell, Profile=qwps
 - New Application**
 - This page provides links to create new applications.
 - Install a New Application**
 - New Enterprise Application** (highlighted with a red box)
 - New Business Level Application
 - New Asset

- __ d. In the **Path to the new application**, with the **Local file system** selected, click **Browse**.

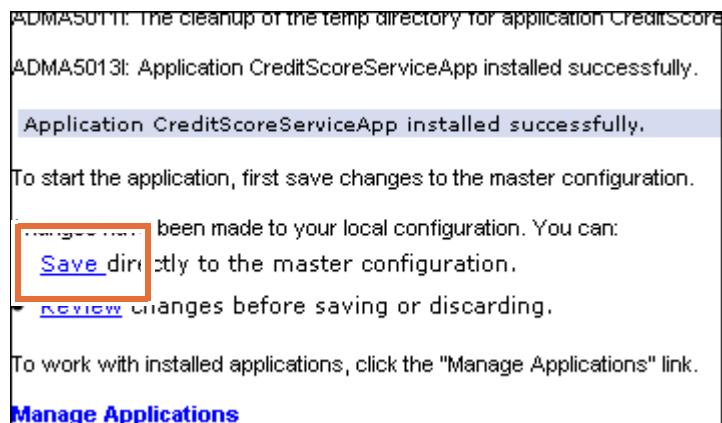
- __ e. Browse to C:\labfiles\Support Files\EX2\CreditScoreService.ear and click Open.



- __ f. Click **Next**.
- __ g. In the **How do you want to install the application** page, select **Fast path** and click **Next**.
- __ h. Review the installation options and click **Step 3 Summary**.



- __ i. Click **Finish** to complete the installation.
- __ j. When the installation completes successfully, click **Save** to save the changes to the master configuration.



- ___ k. Return to the application list by selecting **Applications > Application Types > WebSphere enterprise applications**.
- ___ l. Locate the **CreditScoreServiceApp**, and select its check box.
- ___ m. Click **Start**.

When the application successfully starts, the application status for the **CreditScoreServiceApp** changes to a green arrow.

<input type="checkbox"/>	CreditScoreServiceApp	
...		

This installation resolves the issue for the failed invoke that you saw earlier in this lab. You verify that it works by testing the scenario again in the next part of this lab.

- ___ 4. Resubmit the failed events through **Failed Event Manager** to complete the process.



Note

Before you resubmit the events for test, you might want to clear the server logs as you did before.

- ___ a. Clear the server logs in IBM Integration Designer before resubmitting the failed event.
- ___ b. In the administration console, select **Servers > Deployment Environments > ProcessServer > Failed Event Manager**.
- ___ c. Select **Get all failed events** under the **Failed events on this server** section.

- ___ d. Select the check box for the failed event, and click **Resubmit**. If the list contains more than one failed event, then you can click the event of your choice or the same one you clicked earlier in this lab.

Deployment Environments > Failed Event Manager > Search results

The failed events result set shows the failed events found from the most recent search.

Use the buttons below to manage the failed events in the current result set and search.

Preferences

Select	Event ID	Event type	Module	Component
<input checked="" type="checkbox"/>	PI:90030147.834..	BPC	FoundationModule..	Accounting
<input type="checkbox"/>	PI:90030147.836..	BPC	FoundationModule..	Accounting
<input type="checkbox"/>	PI:90030147.836..	BPC	FoundationModule..	Accounting
Total 3				

- ___ e. Verify that the failed event was resubmitted successfully.

<input checked="" type="checkbox"/> Messages
■ Failed event [PI:90030147.834ecb66.a5afef53.e91b00ba] was resubmitted successfully.

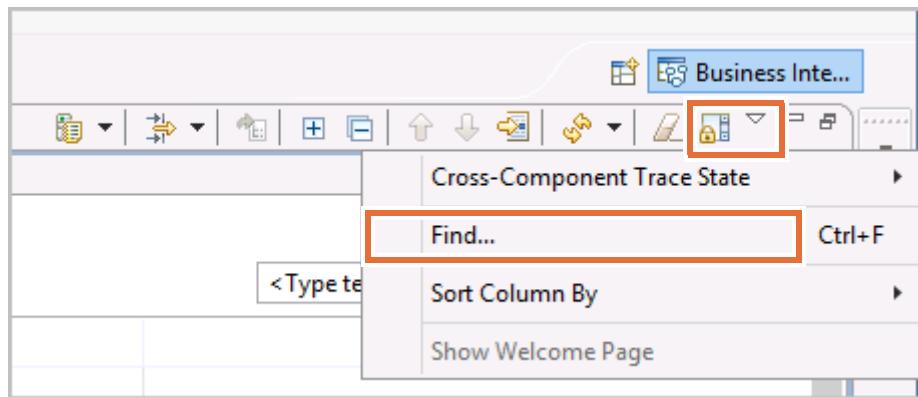
- f. In the **Server Logs** view, review the event that went through successfully as compared to the failed event that you saw earlier. Expand the message unless it is already expanded. Note the successful import and start of the component by scrolling down the logs.

Console (filtered): IBM Process Server v8.5.7 at localhost		
Type	T...	Contents
Invocation sequence	De...	00...
Start BPEL process (AccountVerification)	De...	00... Start of processing for BPEL process AccountVerificat...
Start invoke (CreditScoreServiceImport:calcula	De...	00... Start of the invocation of operation CreditScoreS...
Start import (CreditScoreServiceImport:calcula	De...	00... Start of the import processing of operation CreditScore...
End import (CreditScoreServiceImport:calcula	De...	00... End of the import processing of operation CreditScore...
End invoke (CreditScoreServiceImport:calcula	De...	00... End of the invocation of operation CreditScoreS...
End BPEL process (AccountVerification)	De...	00... End of processing for BPEL process AccountVerificati...
Start invoke (CreditRiskAssessmentImport:Inpu	De...	00... Start of processing for BPEL process AccountVerificati...
Start import (CreditRiskAssessmentImport:Inpu	De...	00... Start of the import processing of operation CreditRiskAssess...
End import (CreditRiskAssessmentImport:Inpu	De...	00... End of the import processing of operation CreditRiskAssess...
End invoke (CreditRiskAssessmentImport:Inpu	De...	00... End of the invocation of operation CreditRiskAssess...
End BPEL process (AccountVerification)	De...	00... End of processing for BPEL process AccountVerificati...
Start BPEL process (AccountVerification)	De...	00... Start of processing for BPEL process AccountVerificati...
Start submit callback (AccountVerification:Inp	De...	00... 07e8a3dd-ddeb-4640-be64-54345fc78d94 BEGIN a52f...
Start callback (AccountVerification:InputCri	De...	00... Start of the call-back invocation of operation Account...
End callback (AccountVerification:InputCri	De...	00... End of the call-back invocation of operation Account...
End submit callback (AccountVerification:Inp	De...	00... 07e8a3dd-ddeb-4640-be64-54345fc78d94 END a52f6...
End BPEL process (AccountVerification)	De...	00... End of processing for BPEL process AccountVerificati...



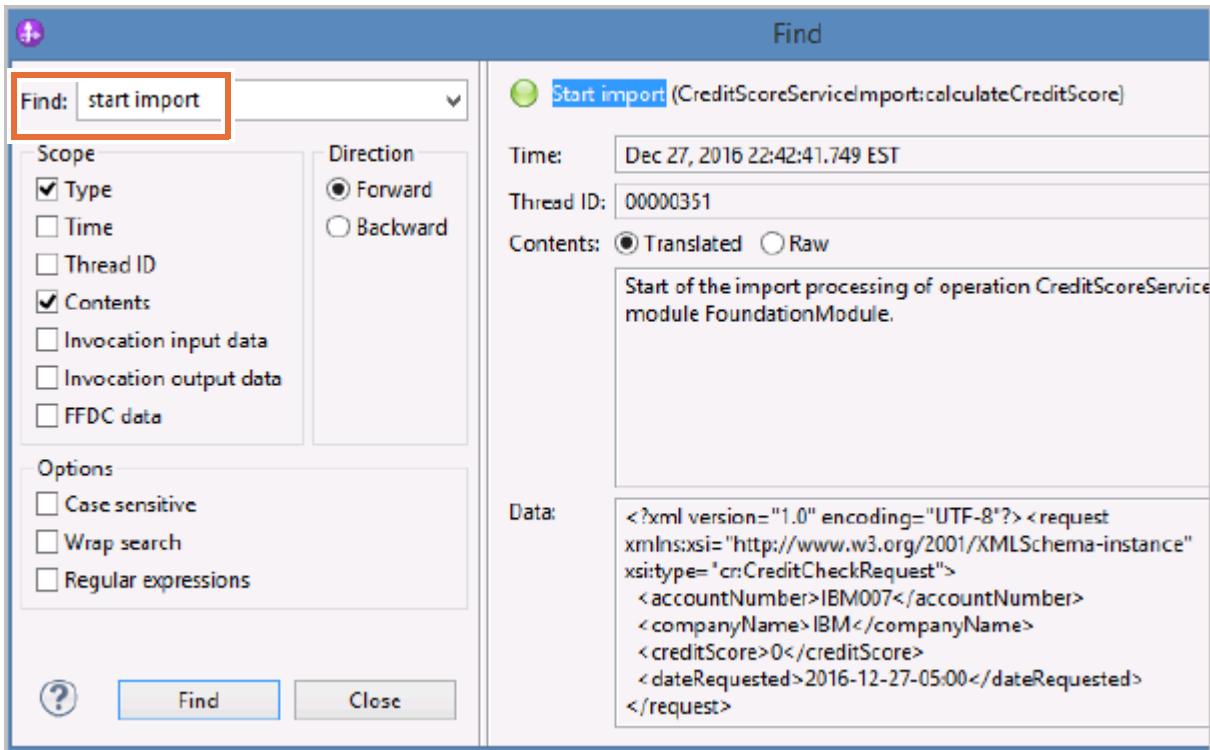
Information

If you cannot locate some item in the output, cross component trace has a find feature that you can use. If you need help searching for the correct import component, click the down arrow icon at the upper right of the **Server Logs** view and click **Find**.



As you can see, several options can be used to search the appropriate data. Enter `start import` and click **Find** to search within the output message. Depending on where your active cursor is in

the logs, you must select **Forward** or **Backward** as the **Direction**. Feel free to explore this feature.



Click **Close** when you are done exploring.

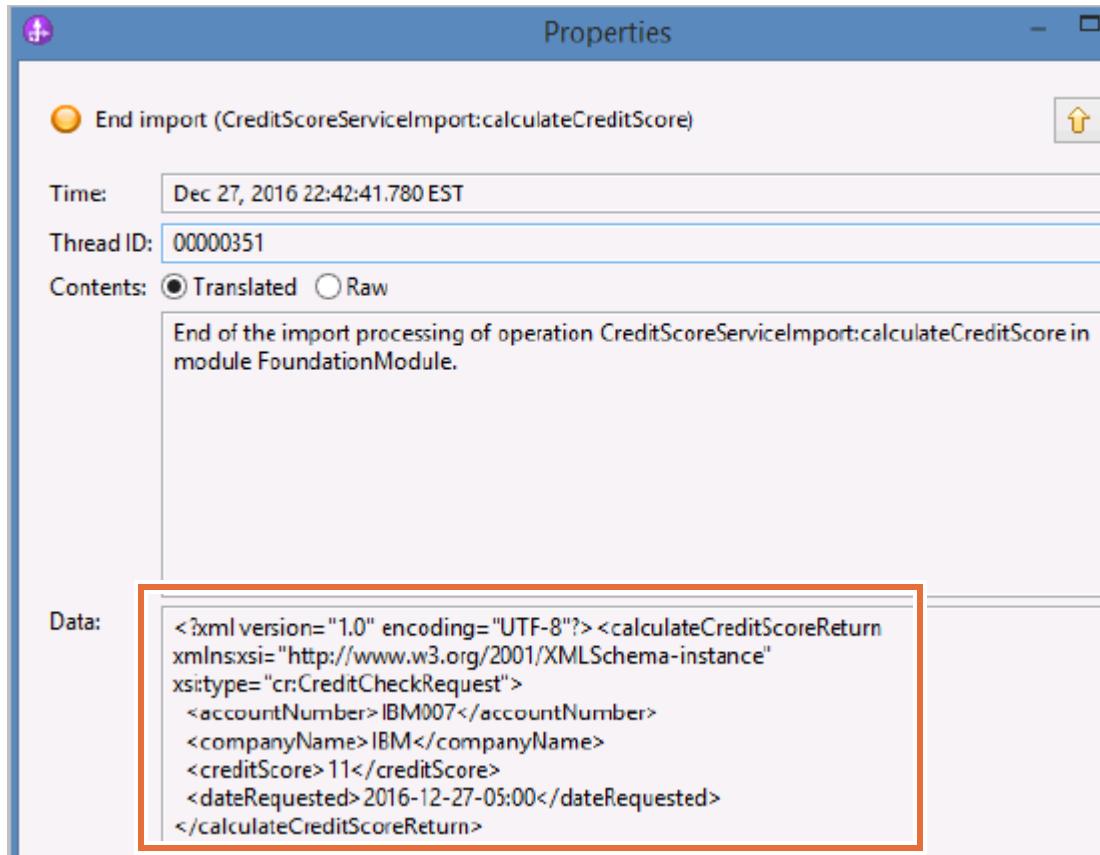
- ___ g. Double-click the **Start import (CreditScoreServiceImport:calculateCreditScore)** message in the **Server Logs** view.

- h. Examine the message contents.

The screenshot shows a software window titled "Properties". At the top left is a green circular icon with a white play symbol. To its right is the title "Properties". On the far right are standard window control buttons for minimize, maximize, and close. Below the title bar, there is a status bar with a green circle icon followed by the text "Start import (CreditScoreServiceImport:calculateCreditScore)". To the right of this status bar is a yellow square icon with a blue arrow pointing upwards. The main content area contains several sections: "Time:" with the value "Dec 27, 2016 22:42:41.749 EST"; "Thread ID:" with the value "00000351"; and "Contents:" with two radio button options: "Translated" (selected) and "Raw". Below these options is a text area containing the message body: "Start of the import processing of operation CreditScoreServiceImport:calculateCreditScore in module FoundationModule.". At the bottom left of the content area is a "Data:" label. To its right is a large text box containing XML code, which is highlighted with a red rectangular border. The XML code is as follows:

```
<?xml version="1.0" encoding="UTF-8"?> <request  
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
xsi:type="cr:CreditCheckRequest">  
    <accountNumber>IBM007</accountNumber>  
    <companyName>IBM</companyName>  
    <creditScore>0</creditScore>  
    <dateRequested>2016-12-27-05:00</dateRequested>  
</request>
```

- i. Click the down arrow to view the **End Import** event property. The **End import (CreditScoreServiceImport:calculateCreditScore)** shows the **creditScore** of **11** instead of **0**.



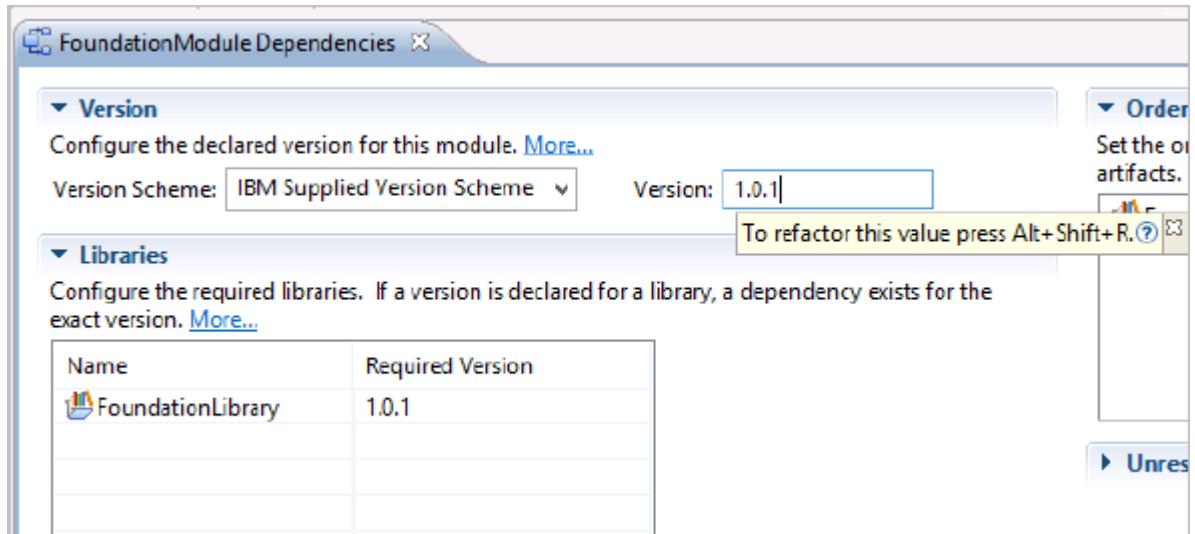
- j. Click **OK**.

Part 6: Create a second version of the module and library

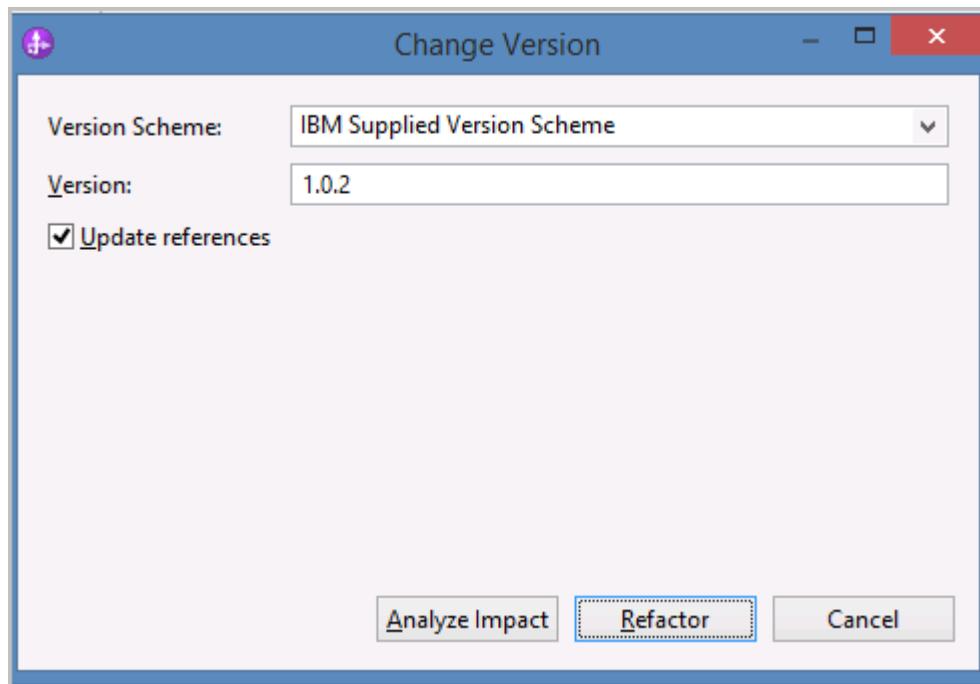
In this section, you create a version of **FoundationModule** and **FoundationLibrary**. Then, you deploy them and verify the version information.

- 1. Change the version number of **FoundationModule** to **1.0.2**.
 - a. In the Business Integration view, expand **FoundationModule** and double-click **Dependencies**.
 - b. In the **FoundationModule dependencies** editor, expand the **Version** section to update the current version of this module.

- __ c. Click inside the **Version** field and press **Alt+Shift+R** to refactor it.



- __ d. In the **Change Version** window, change the **Version** to **1.0.2** and click **Refactor**.



- __ e. Click **OK** to close the **Concurrent Modules** window. Verify that the version of the **FoundationModule** changed to 1.0.2. The current version of **FoundationLibrary** is still 1.0.1.

The screenshot shows the **Version** section with the **Version Scheme** set to **IBM Supplied Version Scheme** and the **Version** field set to **1.0.2**. The **Libraries** section contains a table with one row for **FoundationLibrary**, which has a required version of **1.0.1**.

Name	Required Version
FoundationLibrary	1.0.1

- __ 2. Change the version of **FoundationLibrary** to 1.0.2.
- Expand **FoundationLibrary** and double-click **Dependencies**.
 - Expand the **Version** section.
 - Click inside the **Version** field and press **Alt+Shift+R** to refactor it.

The screenshot shows the **Version** section with the **Version Scheme** set to **IBM Supplied Version Scheme** and the **Version** field set to **1.0.1**. A tooltip above the field says **To refactor this value press Alt+Shift+R.**

- In the **Change Version** window, change the **Version** to 1.0.2 and click **Refactor**.
- Verify that the version is 1.0.2.

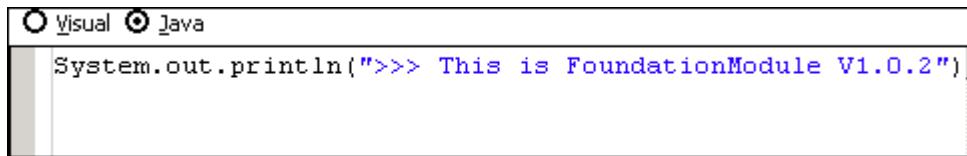
The screenshot shows the **Version** section with the **Version Scheme** set to **IBM Supplied Version Scheme** and the **Version** field set to **1.0.2**.

- ___ f. Switch to the **FoundationModule Dependencies** tab and note that the **Required Version** for **FoundationLibrary** is 1.0.2.

The screenshot shows the 'Version' tab of a module configuration. It has two sections: 'Version' and 'Libraries'. In the 'Version' section, the 'Version Scheme' is set to 'IBM Supplied Version Scheme' and the 'Version' is '1.0.2'. In the 'Libraries' section, there is a table with two columns: 'Name' and 'Required Version'. A single row is present, showing 'FoundationLibrary' in the Name column and '1.0.2' in the Required Version column.

Name	Required Version
FoundationLibrary	1.0.2

- ___ 3. Add a Java snippet to the **AccountVerification** process that shows version information in the server console. The console output is used to verify that version 1.0.2 of the application is the one that is running.
- ___ a. In the Business Integration view, expand **FoundationModule > Integration Logic > BPEL Processes > AccountVerification** and double-click **AccountVerification**.
 - ___ b. Select the **VersionInfoSnippet** activity, switch to the **Properties** view, and select the **Details** tab.
 - ___ c. Change the version number from v1.0.1 to: v1.0.2

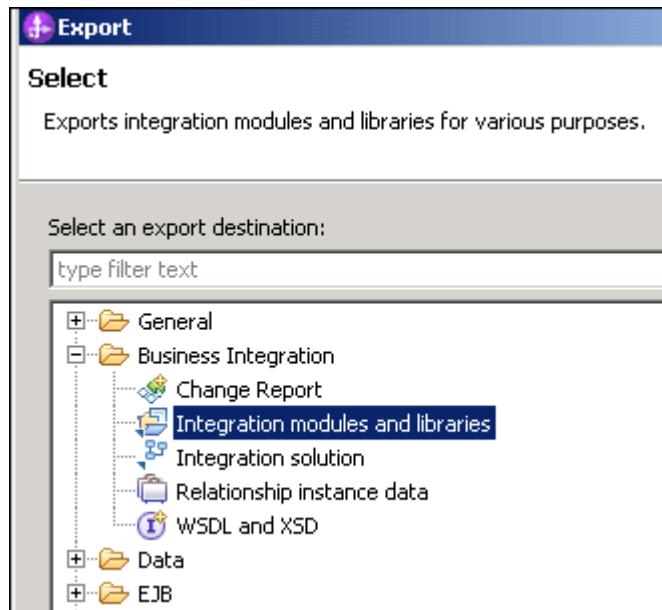


- ___ d. Click an empty portion of the canvas outside the **AccountVerification_Flow** activity.
 - ___ e. Switch to the **Properties** view and click the **Details** tab.
 - ___ f. Scroll down until you see the **Select the date when the process becomes valid** property. Click the **Select Date** icon and change the date to today.
 - ___ g. Change the time by selecting **0** for **Hours**, **0** for **Minutes**, and **0** for **Seconds**.
- ___ 4. Save your changes by pressing Ctrl+S and close all open tabs.

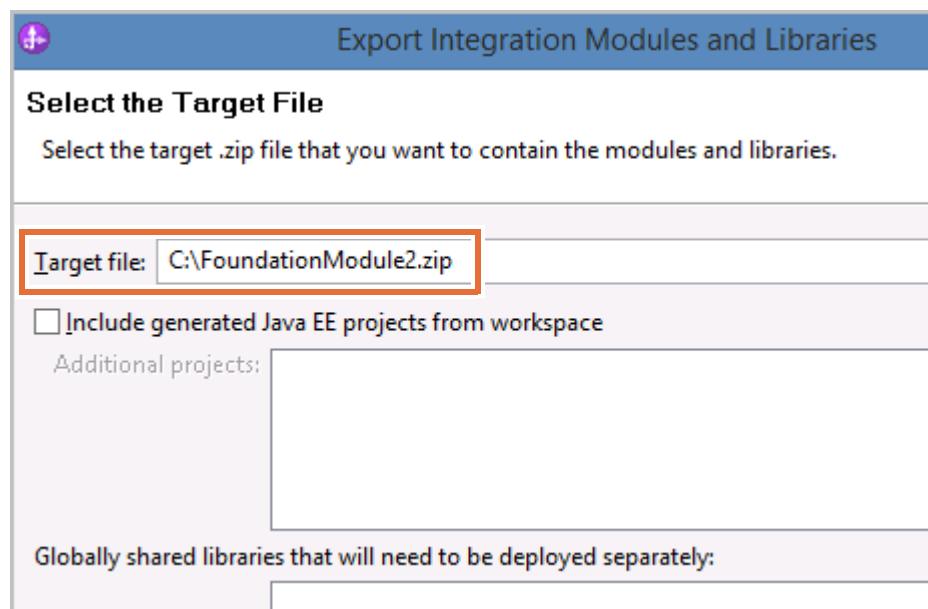
Part 7: Use the serviceDeploy tool to publish the new version of FoundationModule

In this section, deploy the versioned module to the server and verify that the version information is updated.

- 1. Export **FoundationModule** for command-line service deployment.
 - a. In the Business Integration view, right-click **FoundationModule** and select **Export**.
 - b. In the **Export** window, expand **Business Integration** and select **Integration modules and libraries**.

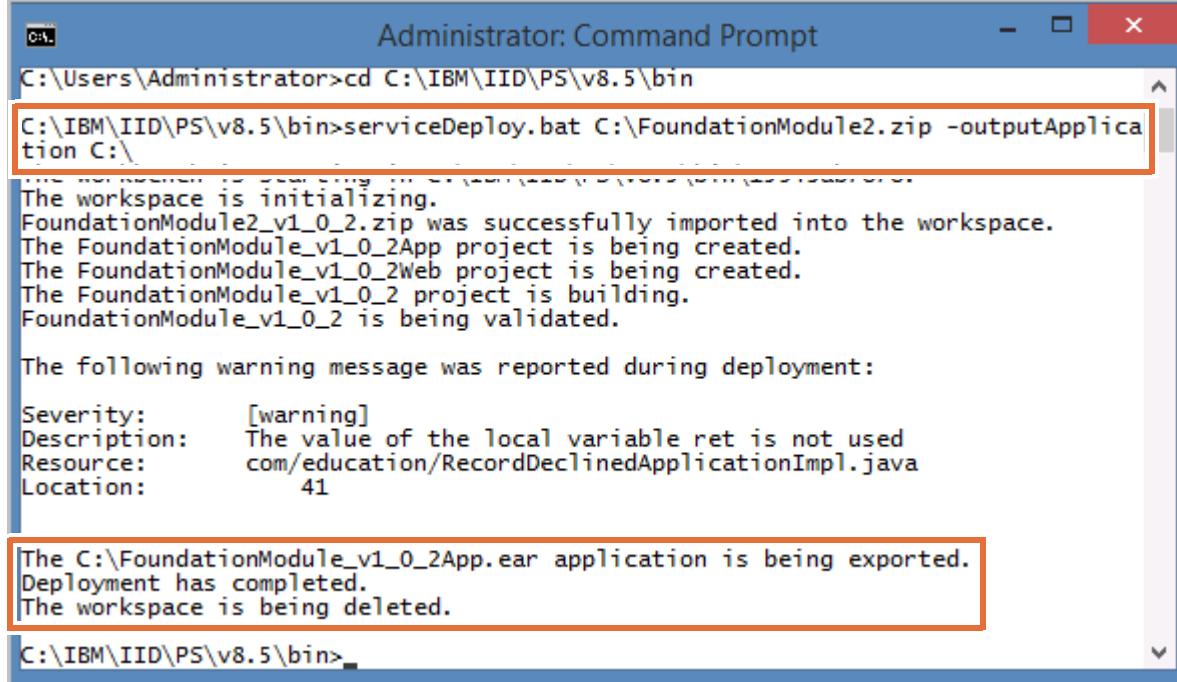


- c. Click **Next**.
- d. In the **Export Integration Modules and Libraries** window, select **Command line service deployment**.
- e. Verify that **FoundationModule** is the only module selected.
- f. Click **Next**.
- g. In the **Target file** field, enter: `C:\FoundationModule2.zip`



- ___ h. Click **Finish**.
- ___ 2. Run the `serviceDeploy` command to create a version-aware EAR file.
- ___ a. On the desktop, right-click the windows logo in the lower-left corner and click **Run**. Enter `cmd` in the **Run** dialog box, and then click **OK** to start a command window. Optionally, you can click the command prompt shortcut in the taskbar.
 - ___ b. Change the working directory by entering `cd C:\IBM\IID\PS\v8.5\bin` and then press the Enter key.
 - ___ c. Enter the following command and press Enter:

```
serviceDeploy.bat C:\FoundationModule2.zip -outputApplication C:\
```



The screenshot shows an 'Administrator: Command Prompt' window. The command entered was `C:\IBM\IID\PS\v8.5\bin>serviceDeploy.bat C:\FoundationModule2.zip -outputApplication C:\`. The output shows the workspace initializing, FoundationModule2_v1_0_2.zip being imported, and projects being created and built. A warning message about a local variable 'ret' not being used is displayed. The final message indicates the application is being exported, deployment is completed, and the workspace is being deleted.

```
C:\Users\Administrator>cd C:\IBM\IID\PS\v8.5\bin
C:\IBM\IID\PS\v8.5\bin>serviceDeploy.bat C:\FoundationModule2.zip -outputApplication C:\

The workspace is initializing.
FoundationModule2_v1_0_2.zip was successfully imported into the workspace.
The FoundationModule_v1_0_2App project is being created.
The FoundationModule_v1_0_2Web project is being created.
The FoundationModule_v1_0_2 project is building.
FoundationModule_v1_0_2 is being validated.

The following warning message was reported during deployment:

Severity: [warning]
Description: The value of the local variable ret is not used
Resource: com/education/RecordDeclinedApplicationImpl.java
Location: 41

The C:\FoundationModule_v1_0_2App.ear application is being exported.
Deployment has completed.
The workspace is being deleted.

C:\IBM\IID\PS\v8.5\bin>
```

- ___ d. After the command runs successfully, verify that the `FoundationModule_v1_0_2App.ear` file is generated under `C:\`. Close the command window.
- ___ 3. Install the versioned SCA module on the server.
- ___ a. If it is not already open, start the administrative console. Start Firefox and enter the following address to open the administrative console:
`https://localhost:9043/ibm/console`

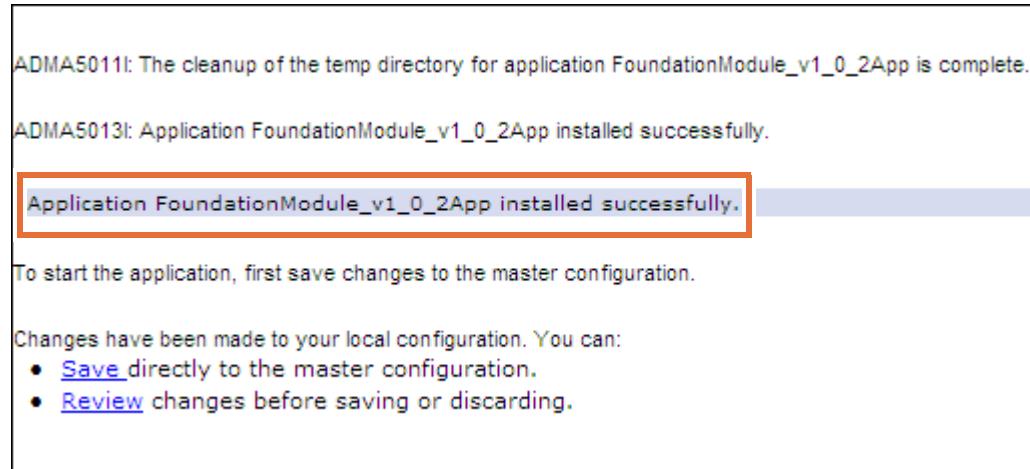


Hint

You can also start the administrative console by right-clicking the server in the **Servers** view and clicking **Administration > Run administrative console**.

-
- ___ b. At the login page, use **User ID** `admin` and **Password** `web1sphere` and click **Log in**.
 - ___ c. Click **Applications > SCA modules**.

- __ d. Click **Install**.
- __ e. Click **Browse**, go to `C:\FoundationModule_v1_0_2App.ear`, and click **Open**.
- __ f. Click **Next**.
- __ g. Wait until the installation completes successfully.



- __ h. Click **Save**.
- __ i. Notice that the `FoundationModule` is installed and the new version information is shown in the **Version** column.

The screenshot shows the **SCA modules** page. At the top, there is a heading **SCA modules** with a sub-instruction: "This page shows all installed Service Component Architecture (SCA) modules and their services. SCA modules encapsulate services, so you can make changes to services without affecting the rest of the system. To use the SCA module services you start the associated application." Below this is a toolbar with buttons: Start, Stop, Install, and Uninstall. Underneath is a table with columns: Select, Module, Version, and Application. The table lists the following resources:

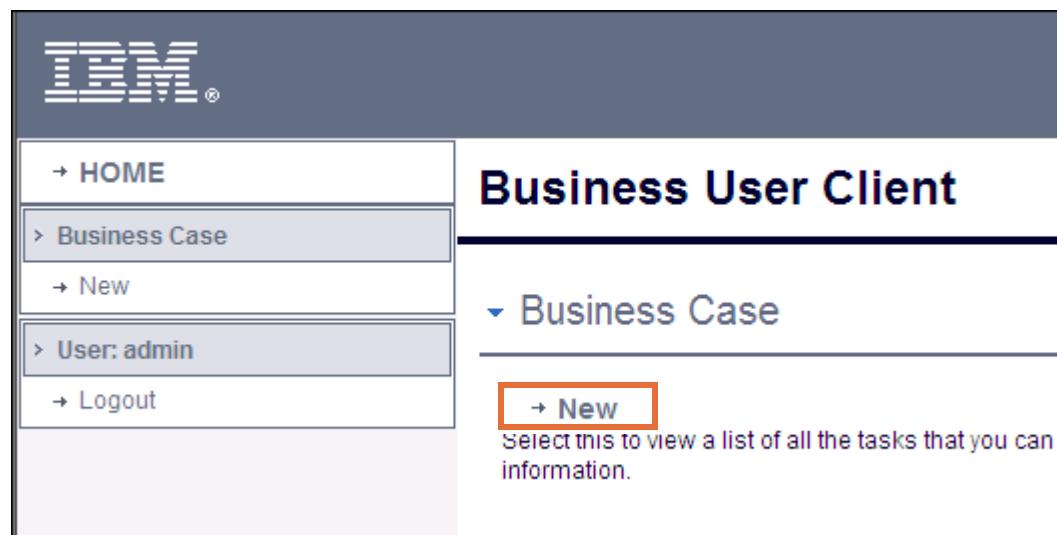
Select	Module	Version	Application
<input type="checkbox"/>	BFMIF_Node1_server1		BPEContainer_Node1_server1
<input type="checkbox"/>	CreditScoreService		CreditScoreServiceApp
<input type="checkbox"/>	FoundationModule	1.0.1	FoundationModule_v1_0_1App
<input checked="" type="checkbox"/>	FoundationModule	1.0.2	FoundationModule_v1_0_2App
<input type="checkbox"/>	FoundationServices		FoundationServicesApp

- __ j. Select the check box for `FoundationModule` version 1.0.2 and click **Start**. When the application is started, the **Status** changes to a solid green arrow.
- __ k. Log out of the administrative console.

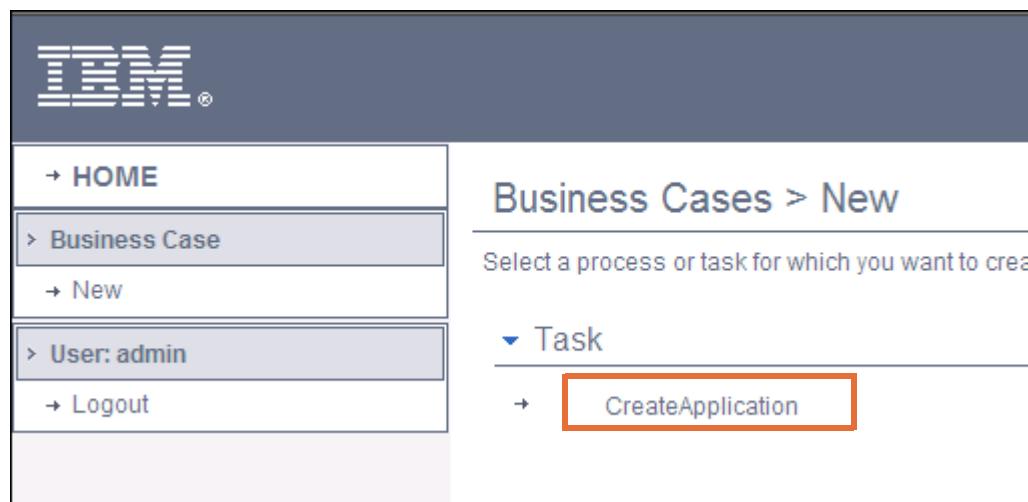
Part 8: Test the versioned application

In this section, run the business application and verify the version information.

- 1. Create an account opening request to test the **Create Application** task user interface JSP.
 - a. In Firefox and type the following address in the URL:
`https://localhost:9443/AccountOpeningUI/Index.jsp`
 The login page for the account creation user interface opens.
 - b. At the login page, use **Name** `admin` and **Password** `web1sphere` and click **Log in**.
 - c. On the **Business User Client** page, click **New**.



- d. **CreateApplication** is the only available task. Click that link.



- e. Switch to the **Server Logs** view in IBM Integration Designer, and clear the logs before you start the test. In the menu bar in the **Server Logs** view, click the clear server console icon.

- ___ f. Switch back to the **CreateApplication** page in the browser and enter `IBM` in the **companyName** field. Leave the rest of the fields blank and click **Create**.

Business Cases > New > CreateApplication

Enter the values for the input data and optionally provide additional information

▼ Input Data

accountNumber	<input type="text"/>
applicationDate	<input type="text"/>
applicationDecision	<input type="checkbox"/>
comments	<input type="text"/>
companyName	<input type="text" value="IBM"/>
contactFirstName	<input type="text"/>
contactLastName	<input type="text"/>
contactPhoneNumber	<input type="text"/>
creditRating	<input type="text"/>

- ___ g. Switch back to the **Server Logs** tab. When the AccountVerification process starts, a system out message confirms that version 1.0.2 is being called. You might have to expand the output to view the log message. It is the same text that you added to the **VersionInfoSnippet** in your module.

Console (filtered): IBM Process Server v8.5.7 at localhost

Show All Record Types (Hierarchical) > with only Server State and Error Contents (Page 1 of 1)

Type	T...	Contents
Start component (AccountVerification:InputCriterion)	De...	Start of the component processing of operation...
In BPEL process	De...	00... 07e8a3dd-ddab-4640-be64-54345fc78d94 STA...
Start BPEL process (AccountVerification)	De...	00... Start of processing for BPEL process AccountV...
Log message	De...	>>> This is FoundationModule V1.0.2
Start invoke (DetermineApplicantEligibility:InputCriteri...	De...	Start of the invocation of operation Determine...
Start component (DetermineApplicantEligibility:InputCriteri...	De...	00... Start of the component processing of operatio...
Log message	De...	[Java] Determine Applicant Eligibility - begins
Log message	De...	[Java] Determine Applicant Eligibility - ends
End component (DetermineApplicantEligibility:InputCriteri...	De...	00... End of the component processing of operatio...
End invoke (DetermineApplicantEligibility:InputCriteri...	De...	00... End of the invocation of operation Determine...
End BPEL process (AccountVerification)	De...	00... End of processing for BPEL process AccountVe...

The final test is complete. This test verifies a valid version of the module, which is called at run time. Investigate the cross-component trace log to determine the flow of the business process.

- __ 2. Remove running instances and applications from the server.
 - __ a. In Firefox, enter <https://localhost:9443/bpc/> in the URL.
 - __ b. At the login page, use **User Name** admin and **Password** web1sphere and click **Log in**.
 - __ c. Click **All Versions**. Two versions of the **AccountVerification** template are listed.

The screenshot shows the BPC interface. On the left, there's a sidebar with tabs for 'Views' and 'Reports'. Under 'Views', there are sections for 'Process Templates' (with 'Currently Valid' and 'All Versions' options) and 'Process Instances' (listing 'Started By Me', 'Administered By Me', 'Critical Processes', 'Terminated Processes', and 'Failed Compensations'). On the right, a main panel titled 'All Versions Of Template' is displayed. It says 'Use this page to view all versions'. Below that are buttons for 'Instances' and 'View Structure'. A list shows 'Process Template Name' with two entries: 'AccountVerification' and 'AccountVerification'. At the bottom, it says 'Items found: 2 Items selected:'.

- __ d. Click **Administered By Me** under **Process Instances**. If a running process instance is listed, select the check box next to the running instance and click **Terminate**. Again, select the check box next to the running instance and click **Delete**. If no running instance is listed under **Administered By Me**, you can ignore this step.
- __ e. Click **All Tasks** under **Task Instances** to examine the running instances of human tasks.

The screenshot shows the BPC interface. On the left, there's a sidebar with tabs for 'Views' and 'Reports'. Under 'Views', there are sections for 'Activity Instances' (with 'Stopped Activities') and 'Task Instances' (with 'My To-dos', 'Initiated By Me', 'Administered By Me', and 'My Escalations'). The 'Task Instances' section has 'All Tasks' selected. On the right, a main panel shows 'Task Templates' with 'My Task Templates'.

- __ f. Click **CreateApplication** in the first row.

The screenshot shows the 'All Tasks' interface. At the top, there are buttons for 'Work on', 'Release', 'Transfer', 'Start', 'Change Business Category', and 'Refresh'. Below these are filter dropdowns for Priority, Task Name, State, Kind, Owner, and Originator. A table lists three task instances:

	Task Name	State	Kind	Owner	Originator
<input type="checkbox"/> 5	CreateApplication	Finished	Invocation Task	admin	
<input type="checkbox"/> 5	CreateApplication	Running	Invocation Task	admin	
<input type="checkbox"/> 5	CreateApplication	Finished	Invocation Task	admin	

At the bottom, it says 'Items found: 3 Items selected: 0' and has navigation buttons '<< Page 1 of 1 >> Item'.

- __ g. Click **Delete** to delete the instance.

The screenshot shows the 'Task Instance' page for a 'CreateApplication' task. It has buttons for 'Delete', 'Restart', 'Reschedule', and 'Change Business Category'. Below is a table with task details:

Task Description	
Task Name	CreateApplication
Description	
Reason	Starter, Administrato

Follow the previous steps to delete any remaining instances.



Troubleshooting

Any task that is in the **Running** state must be terminated before it can be deleted. Select the task and click **Terminate**.



Important

To properly stop and uninstall the application, all process and human task instances must also be stopped. There must not be any running tasks before proceeding.

- __ 3. Start the administrative console and log in.
__ a. Click **Applications > SCA modules**.

- ___ b. Select the check boxes for both versions 1.0.1 and 1.0.2 of **FoundationModule**, and click **Stop** to stop the applications.



- ___ c. After the applications stop, select the check boxes again for both versions 1.0.1 and 1.0.2 of **FoundationModule** and click **Uninstall**.

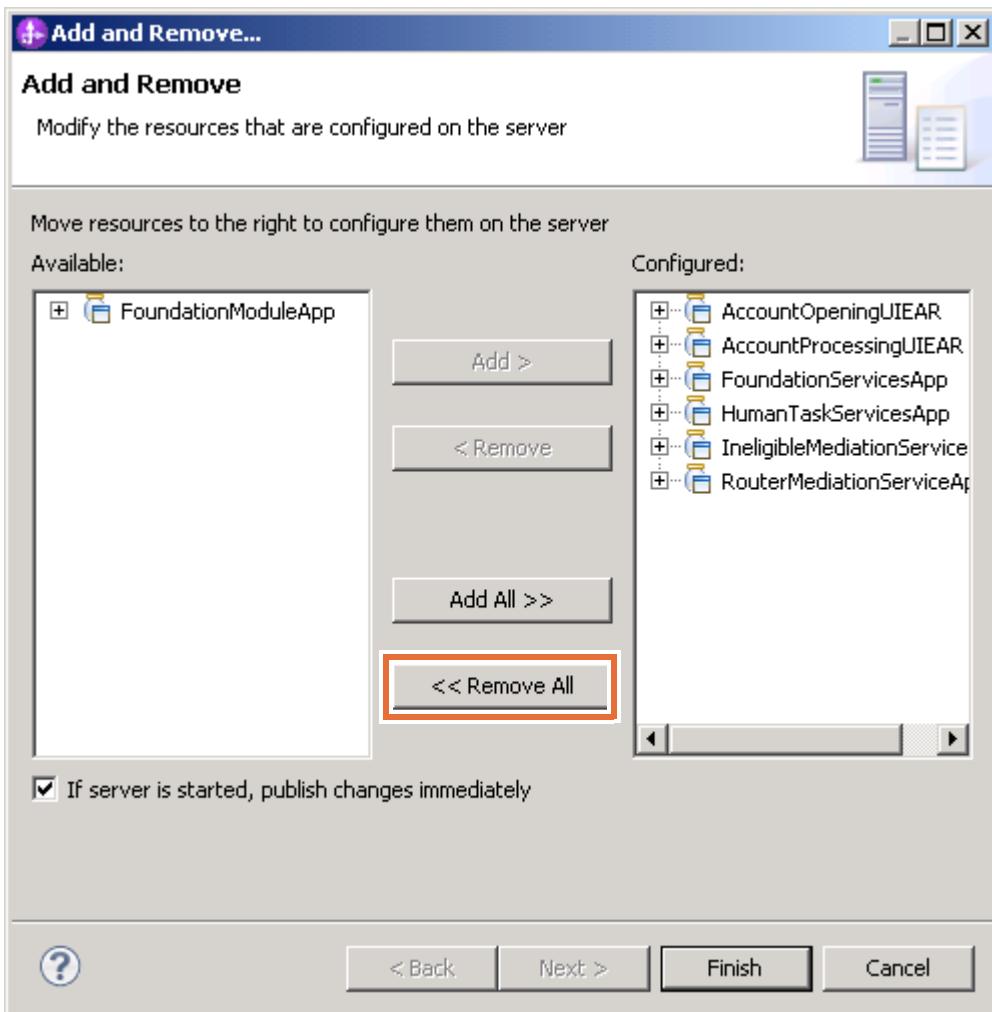


- ___ d. Click **OK** to confirm the uninstall.
 ___ e. Click the **Save** link to save to the master configuration.

Part 9: Clean up the environment

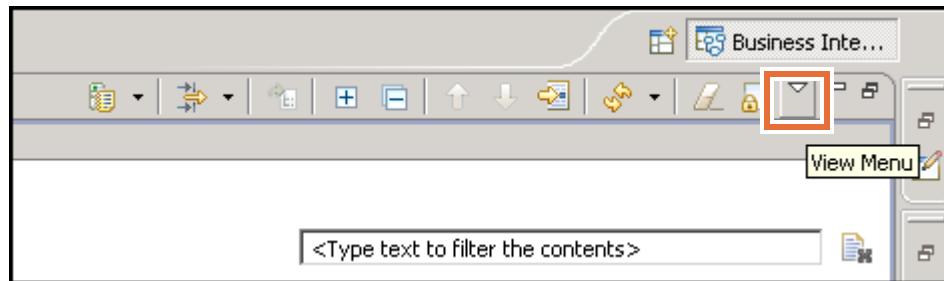
- ___ 1. Remove all of the deployed projects from the server.
 ___ a. In the **Servers** view of Integration Designer test environment, right-click **IBM Process Server v8.5.7 at localhost**, and select **Add and Remove** from its menu.

- ___ b. Click **Remove All**.

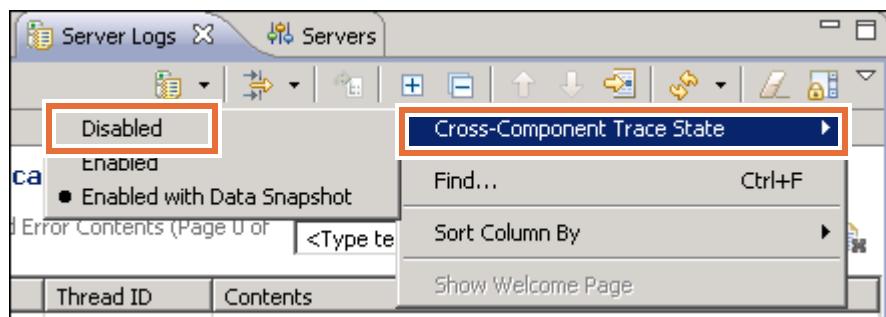


- ___ c. Click **Finish** to complete the uninstallation.
- ___ 2. Uninstall the **CreditScoreService** web service.
- In the administration console, go to **Applications > Application Types > WebSphere enterprise applications**.
 - Select the check box for **CreditScoreServiceApp**, and click **Stop**.
 - When the application is stopped, select the check box for the **CreditScoreServiceApp** again, and click **Uninstall**.
 - Click **OK** at the **Uninstall Application** confirmation page.
 - When the application is successfully uninstalled, you see a message to save the change to the master configuration. Click **Save** to save the configuration.
 - Click **Logout** to log out of the administration console.
- ___ 3. Close all open browser windows.

- ___ 4. Disable cross component trace.
 - ___ a. Access cross component trace settings by clicking the **View menu** arrow at the upper-right corner of the **Server Logs** view.



- ___ b. Disable cross component trace by selecting **Cross-Component Trace State > Disabled** from the View menu.



- ___ 5. Close the IBM Integration Designer. Click **File > Exit**, or click **X** in the upper-right corner.
- ___ 6. Close any other open windows on the desktop.

End of exercise

Exercise 3. Working with SCA bindings and qualifiers

Estimated time

01:00

Overview

In this exercise, you implement a WebSphere MQ import binding in an existing SCA module. You work with a SCA QoS qualifier in this exercise; a subsequent exercise introduces more qualifiers. Finally, you add and configure the WebSphere Adapter for Flat Files to allow data to be read from an input file.

Objectives

After completing this exercise, you should be able to:

- Implement a WebSphere MQ import binding in IBM Integration Designer
- Use the WebSphere Adapter for Flat Files to implement an EIS import binding
- Test bindings in the IBM Integration Designer integrated test environment

Introduction

Messaging providers are equipped with application programming interfaces (APIs), which allow applications to use the services of the messaging provider. With IBM Integration Designer, you can use a number of messaging service providers, both commercially available and custom-written with the use of resource adapters. A resource adapter uses the APIs associated with a messaging provider to allow messages to be passed between the application and the messaging provider.

One such commercially available messaging provider is WebSphere MQ, a family of products that support high-throughput, highly reliable asynchronous messaging services across disparate applications, systems, and platforms. WebSphere MQ provides two sets of APIs for great flexibility in interacting with it from an application: the WebSphere MQ JMS interface and the queuing (MQI). From an IBM Process Server application, you can interact with WebSphere MQ messaging by using either the WebSphere MQ JMS API, or the MQI API. Often you use the WebSphere MQ JMS API for SCA applications, with the use of a resource adapter, which connects the application and WebSphere MQ through JMS messaging. You can also use the native MQI API from an SCA application. A third alternative is to use native JMS, as WebSphere MQ can also be accessed directly through JMS messaging.

Requirements

Completing the exercises for this course requires a lab environment. The environment includes the exercise support files, Mozilla Firefox, IBM Business Process Manager Advanced V8.5.7, IBM DB2 V10.1, RFHUTIL, WebSphere MQ V8.0, and the IBM Process Server V8.5.7 test environment.

Exercise instructions

In this exercise, you extend the existing account scenario. The customer support department must be notified when an application is declined. To implement it, you modify the existing application process to send a message to the customer support system through WebSphere MQ. You also use WebSphere MQ utilities to review the enqueued messages. In the scenario, a corresponding application would receive those messages from WebSphere MQ and process them. You do not construct that receiving system.

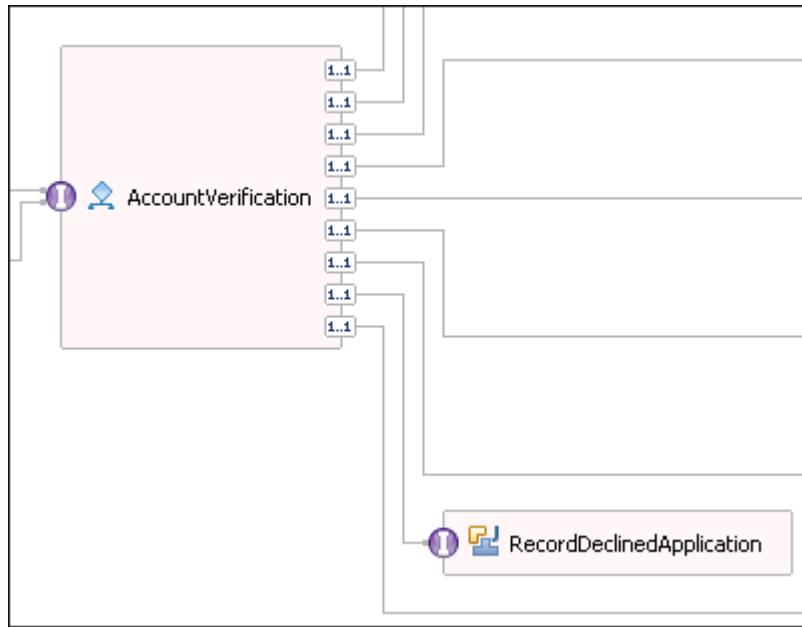
Implementing a WebSphere MQ bound import component

The workspace already contains the end-to-end solution for the account verification process. You extend that by adding a WebSphere MQ import to send a message when a credit application is denied.

Part 1: Adding the WebSphere MQ import

To add the WebSphere MQ import component:

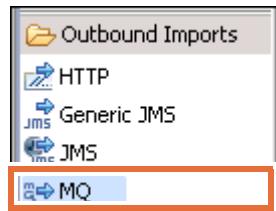
- ___ 1. Open the Exercise 3a workspace.
 - ___ a. On your desktop, open the folder that is labeled **Exercise Shortcuts**.
 - ___ b. Double-click the shortcut that is labeled **Exercise 3a**. IBM Integration Designer starts.
 - ___ c. If you are prompted with the **Automatically Publish** window, click **OK**.
 - ___ d. Wait for the workspace build to complete; it might take a few minutes.
 - ___ e. Close the **Getting Started** tab.
- ___ 2. Examine the assembly diagram for the **FoundationModule**.
 - ___ a. In the **Project** section of the Business Integration view, expand **FoundationModule**.
 - ___ b. Double-click **Assembly Diagram**. The assembly diagram opens in the editor pane.



Note the RecordDeclinedApplication Java component. This component writes information about declined applications to a flat file. You add an import to the assembly diagram that sends the same information to a WebSphere MQ queue.

- ___ 3. Add the WebSphere MQ import to the diagram.
 - ___ a. In the assembly editor **Palette**, expand the **Outbound Imports** drawer.

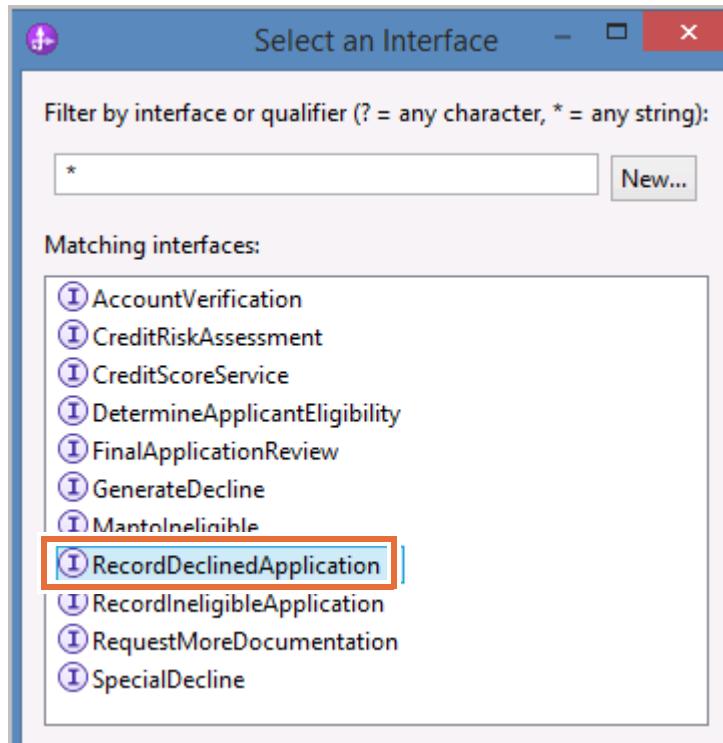
- __ b. Click the **MQ** import.



Important

Be sure to select **MQ**, not MQ JMS.

- __ c. Click the drawing canvas. The **New MQ Import Service** window opens.
- __ d. At the **Select an Interface** window, to the right of the **Interface** field, click **Browse**. The **Select an Interface** window opens.
- __ e. Select **RecordDeclinedApplication** and click **OK**.



- __ f. Click **Next**. The Configure MQ Import Binding window opens.

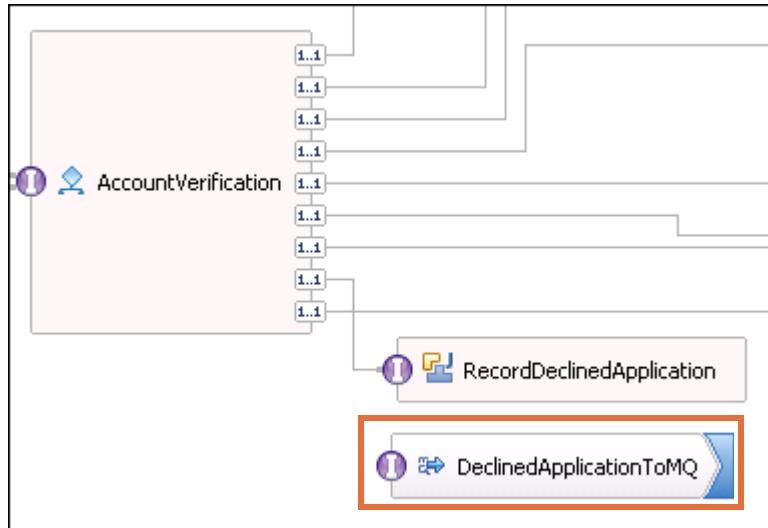
__ g. Enter the following options in the **Configure MQ Import Binding** window:

- Leave the **Specify a configuration view option** field set to `Specify the properties to use to configure WebSphere MQ resources`.
- In the **Request Queue Manager** field, enter: `QM_xpbase`
It is the name of the queue manager that controls the destination queue to which the message is sent.
- In the **Send Queue** field, enter: `DeclinedApplication`
It is the name of the destination queue.
- Leave the **Transport** field set to `Client`.
- In the **Host name** field, enter: `localhost`
It is the name of system where the queue manager is located.
- Leave the **Server channel** field set to `SYSTEM.DEF.SVRCONN`.
It is the channel that is used for WebSphere MQ communication.
- Leave the **port** set to `1414`.
- Leave the **DefaultrequestDataFormat** set to `UTF8XMLDataHandler`.

End-Point Configuration	
Specify a configuration view option:	
<input checked="" type="radio"/> Specify the properties to use to configure WebSphere MQ resources	
<input type="radio"/> Specify the JNDI name for the pre-configured WebSphere MQ resources	
JNDI name for MQ connection factory:	
JNDI name for send queue:	
Request queue manager:	* QM_xpbase
Send queue:	* DeclinedApplication
Transport:	Client
CCDT file:	
Host name:	* localhost
Server channel:	* SYSTEM.DEF.SVRCONN
Port:	* 1414
Data Format	
Default request data format:	UTF8XMLDataHandler

__ h. Click **Finish**. The MQ import is added to the canvas. You might want to move it nearer to the RecordDeclinedApplication component.

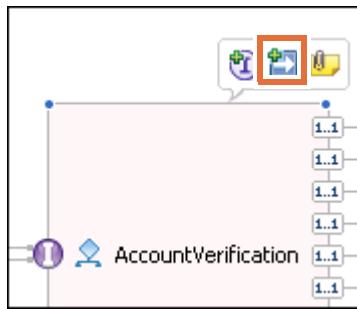
- ___ 4. Change the **Name** of the MQ import to `DeclinedApplicationToMQ`.
 - ___ a. With the **MQImport1** component selected, switch to the **Description** tab in the **Properties** view.
 - ___ b. Change the value in the **Name** field to: `DeclinedApplicationToMQ`



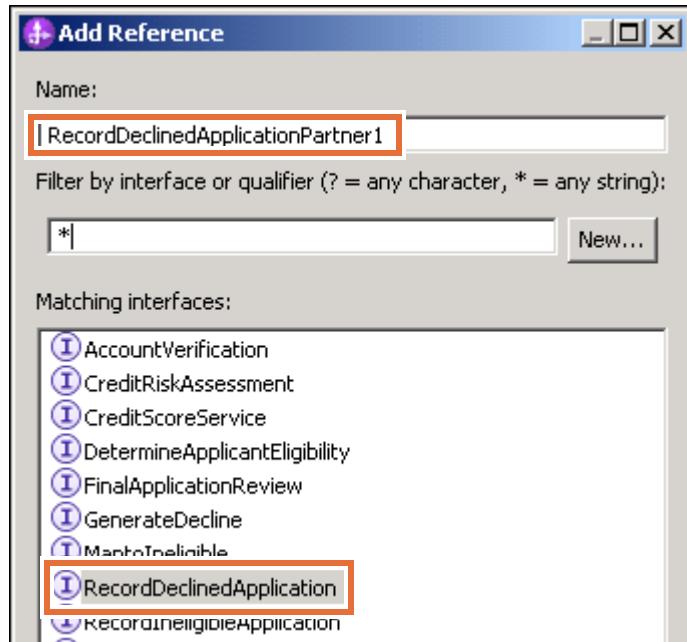
Note

You can also click the import name on the assembly diagram and type over the label.

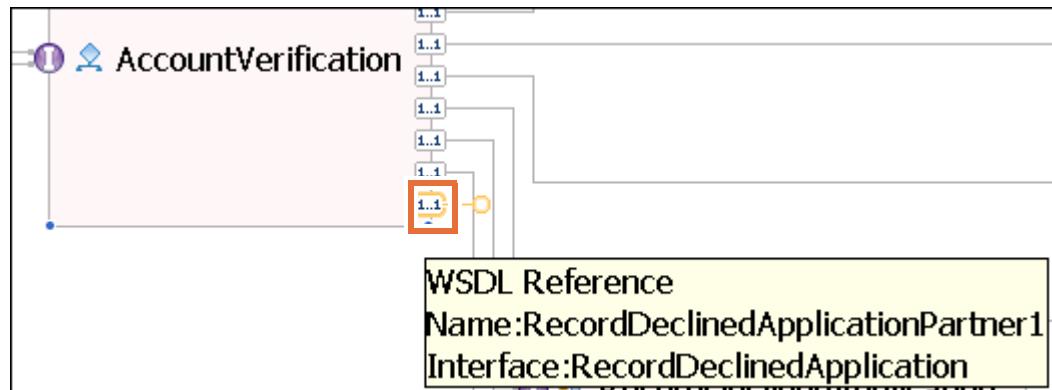
-
- ___ c. Save your workspace (**Ctrl+S**).
 - ___ 5. Add a reference to the AccountVerification process component named `RecordDeclinedApplicationPartner1` that corresponds to the new MQ import.
 - ___ a. Click the **AccountVerification** component.
 - ___ b. Click the **Add Reference** icon in the pop-up action bar.



- ___ c. In the **Add Reference** window, select **RecordDeclinedApplication** under Matching Interfaces. Notice that the Name field at the top of the window shows **RecordDeclinedApplicationPartner1** as the default name because it already has a **RecordDeclinedApplicationPartner** reference.



- ___ d. Click **OK**. The reference is added to the bottom of the AccountVerification component.
- ___ 6. Wire the **DeclinedApplicationToMQ** import to the AccountVerification process component.
- ___ a. Hover the cursor over the **RecordDeclinedApplicationPartner1** reference on the AccountVerification component. An orange, circular wiring handle is shown.



- ___ b. Click and drag the wiring handle onto the interface of the **DeclinedApplicationToMQ** import.



Note

Alternatively, you can right-click the import and select **Wire to Existing** from the menu.

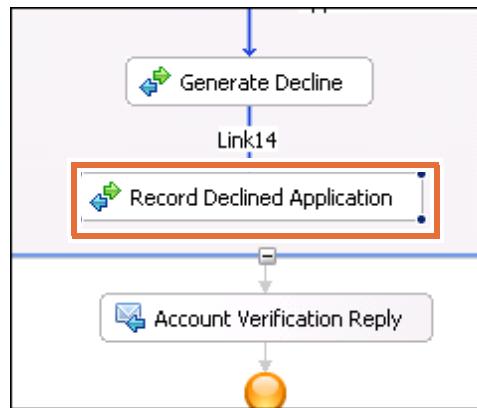
- ___ c. Save the workspace (**Ctrl+S**). Ignore any errors in the Problems view. They are addressed in the next step.
- ___ 7. Synchronize the new partner reference with the BPEL implementation. It adds a BPEL partner reference to the business process that calls the interface of the IBM MQ import that you added to the assembly diagram.
 - ___ a. Right-click the **AccountVerification** process component.
 - ___ b. From the menu, select **Synchronize Interfaces and References > To Implementation**.
 - ___ c. In the **Confirm Synchronize to Implementation** window, click **Yes**.

Part 2: Updating the AccountVerification process component

In the previous part of the exercise, you added a WebSphere MQ import to the FoundationModule assembly diagram. In this step, you add an invoke activity to the AccountVerification business process that calls the WebSphere MQ import. This invocation places a message on a WebSphere MQ queue, which indicates that an application is declined.

To update the AccountVerification process:

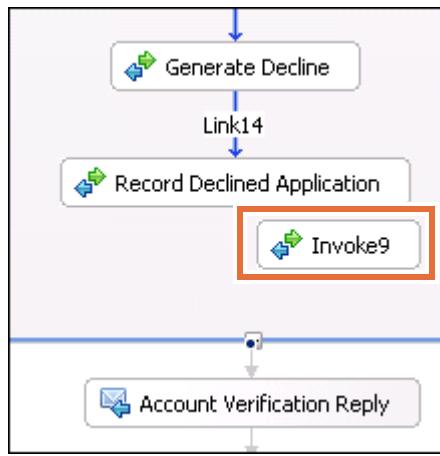
- ___ 1. On the FoundationModule assembly diagram, double-click the **AccountVerification** process component. The process opens in the BPEL editor.
- ___ 2. Scroll to the bottom of the editor pane so that the RecordDeclinedApplication invoke activity is visible.



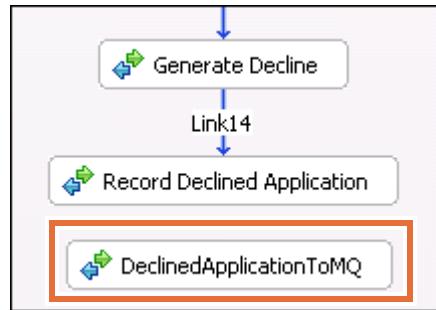
You insert the invoke activity for the IBM MQ import after RecordDeclinedApplication.

- ___ 3. In the **Palette**, expand the **Basic Actions** drawer and click **Invoke**.

- ___ 4. Position the cursor below the RecordDeclinedApplication activity and click to add the activity to the diagram.



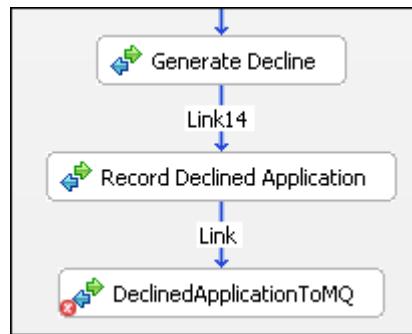
- ___ 5. Rename the newly added invoke activity to DeclinedApplicationToMQ.
- ___ a. Select the new invoke activity.
- ___ b. Switch to the **Description** tab in the **Properties** view.
- ___ c. Change the **DisplayName** to: DeclinedApplicationToMQ



Notice that the **Name** and **Display Name** fields do not have to be synchronized.

- ___ d. Save the workspace.
- ___ 6. Wire the **DeclinedApplicationToMQ** invoke activity to the **Record Declined Application** activity.
- ___ a. Right-click the **Record Declined Application** activity and choose **Add a link** from the menu.
- ___ b. Click the **DeclinedApplicationToMQ** invoke activity to create the link.
- ___ c. Save the workspace. You can ignore any errors in the Problems view.

- ___ d. For readability, right-click inside the AccountVerification generalized flow and choose **Arrange Parallel Activities Contents** from the menu.



- ___ 7. Configure the MQ invoke activity to use the **RecordDeclinedApplicationPartner1** reference partner.
- Select the **DeclinedApplicationToMQ** invoke activity.
 - Switch to the **Properties** view.
 - Click the **Details** tab.
 - Click **Browse** to the right of the **Partner** field.
 - In the **Select a Partner** window, select **RecordDeclinedApplicationPartner1**, and click **OK**. Associating the partner causes a number of other fields that are updated automatically, including the Interface and Operation fields.
 - Verify that the **Interface** is set to **RecordDeclinedApplication**, **Operation** is set to **InputCriterion**, and **Use data type variables mapping** is checked.
 - Click the **none** link in the **Read from Variable** column.

Partner: *	<input checked="" type="checkbox"/> RecordDeclinedApplicationPartner1	Browse...	
Interface: *	RecordDeclinedApplication		
Operation: *	<input checked="" type="checkbox"/> InputCriterion		
<input checked="" type="checkbox"/> Use data type variables mapping			
	Name	Type	Read from Variable
	Input	CustomerApplication	(none) ▼

A list of variables opens.

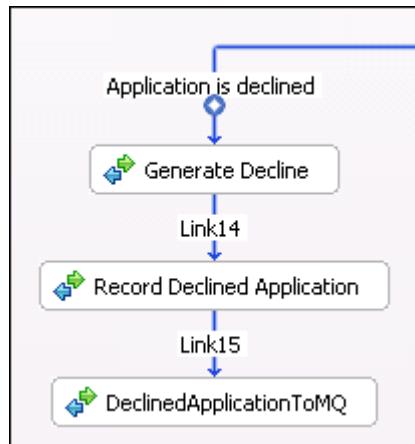
- ___ h. Choose **CustomerApplicationVariable** from the list to update the Read from Variable field.

	Name	Type	Read from Variable
	Input	CustomerApplication	CustomerApplicationVariable ▼

- ___ 8. Save the workspace (**Ctrl+S**). The Problems view must contain no errors. The module is ready for testing.

Part 3: Testing the module

You test the module by using a customer that represents a high credit risk. Because of this risk, the application is declined. The process flow takes the “Application is declined” path, which calls the Generate Decline and Record Declined Application activities. Then, the DeclinedApplicationToMQ activity is called to send the declined application to a WebSphere MQ queue.



To test the module:

- 1. If IBM Process Server is not already running, start it.
 - a. In the **Servers** view, right-click **IBM Process Server v8.5.7 at localhost** and choose **Start** from the menu.



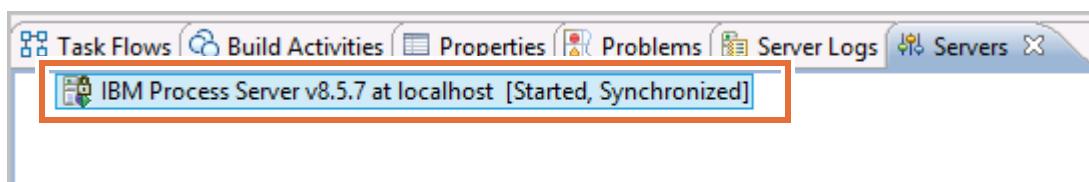
Note

You can also start the server by clicking the **Start** icon in the toolbar.

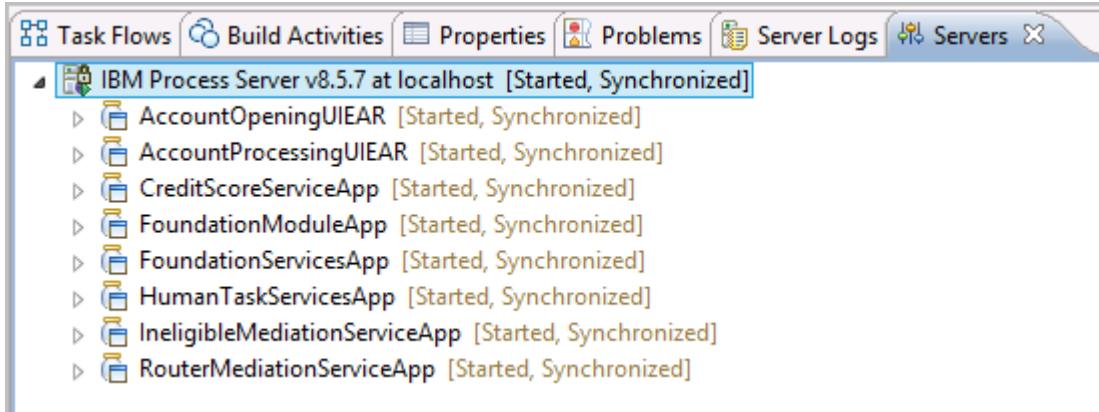


If the server start status shows for more than 5 minutes and the **Server Logs** view is not getting updated with server start messages, then you need to start it manually. Right-click **IBM Process Server v8.5.5 at localhost** and select **Stop** from the menu to stop successfully (do the Stop operation one more time if the server status does not show as Stopped). Restart the server by right-clicking **IBM Process Server v8.5.7 at localhost** and selecting **Start**.

-
- b. Wait for the server to start before proceeding. The **State** column in the **Servers** view shows **Started**, and the **Server Logs** view shows the message **Server server1 open for e-business**.



- __ 2. Publish the applications to the server.
 - __ a. In the **Servers** view, right-click **IBM Process Server v8.5.7 at localhost** and choose **Add and Remove** from the menu.
 - __ b. Click **Add All** to add the projects to the **Configured** list.
 - __ c. Click **Finish**. The projects are deployed to the server. Wait for the deployment process to complete. When the projects are deployed, each application has a status of Started in the **Status** column.



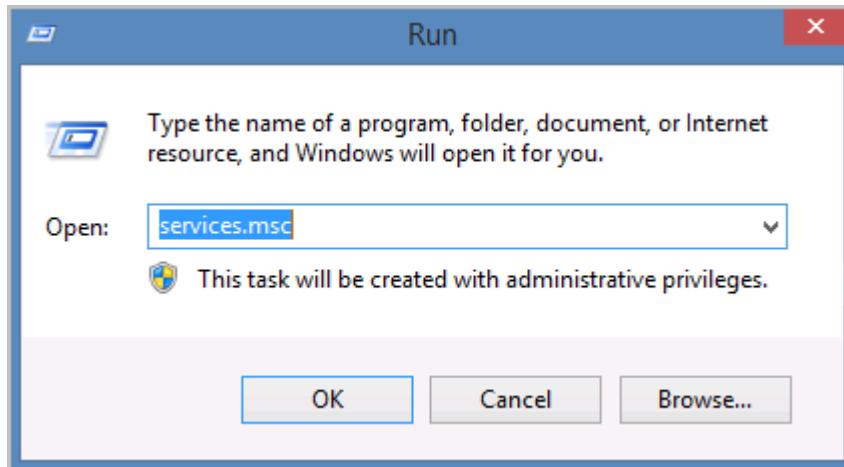
Note

If you see any of the applications not in started state, then in the **Servers** view, right-click **IBM Process Server v8.5.7 at localhost** and select **Publish**.

Additionally, in the **Server Logs** view, you see several messages that are indicated to show that the deployed applications started successfully, such as `Application started: FoundationModuleApp`.

- __ 3. Verify that the IBM WebSphere MQ service is started.
 - __ a. On the Windows desktop, right-click the Windows logo icon and click **Run** from the menu.

- ___ b. Enter services.msc in the Run window and click **OK**. The **Services** window opens.



- ___ c. Locate the **IBM WebSphere MQ** service.

Name	Description	Status
Hyper-V Volume Shadow Copy Re...	Coordinates the co...	
IBM Secure Shell Server for Windows	IBM Secure Shell Se...	Running
IBM WebSphere MQ (Installation1)	Provides startup an...	
IKE and AuthIP IPsec Keying Modu...	The IKEEXT service ...	Running
Interactive Services Detection	Enables user notific...	
Internet Connection Sharing (ICS)	Provides network a...	

- ___ d. If the service is not started already, right-click **IBM WebSphere MQ** and choose **Start** from the menu, The service is running when the **Status** column shows **Running**.

Name	Description	Status
Hyper-V Volume Shadow Copy Re...	Coordinates the co...	
IBM Secure Shell Server for Windows	IBM Secure Shell Se...	Running
IBM WebSphere MQ (Installation1)	Provides startup an...	Running
IKE and AuthIP IPsec Keying Modu...	The IKEEXT service ...	Running
Interactive Services Detection	Enables user notific...	

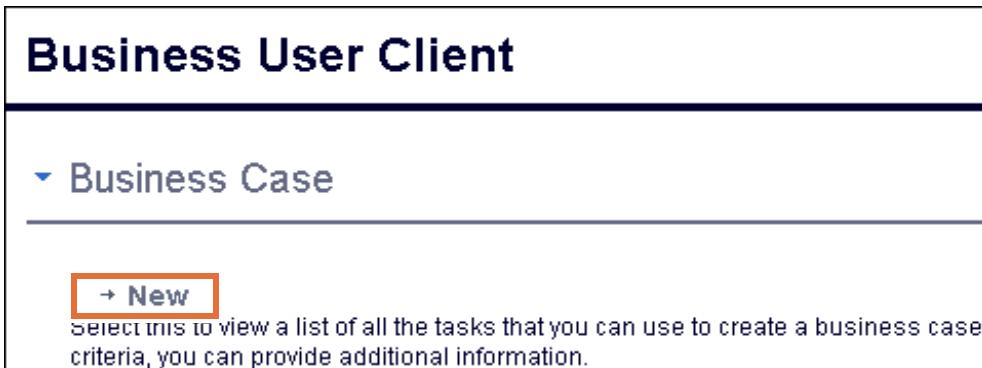
- ___ e. Minimize (but do not close) the Services window. You can use it later to stop the WebSphere MQ service.
4. Enter the initial test data through the AccountOpeningUI human task user interface. You use test data for TestCo. TestCo represents a company that is a high credit risk. As a result, the business process declines the application.
- ___ a. Start Firefox and enter the address:
<https://localhost:9443/AccountOpeningUI/Login.jsp>
The Login to Business User Client web page is opened.

- __ b. Use **Name** admin and **Password** web1sphere and click **Login**.



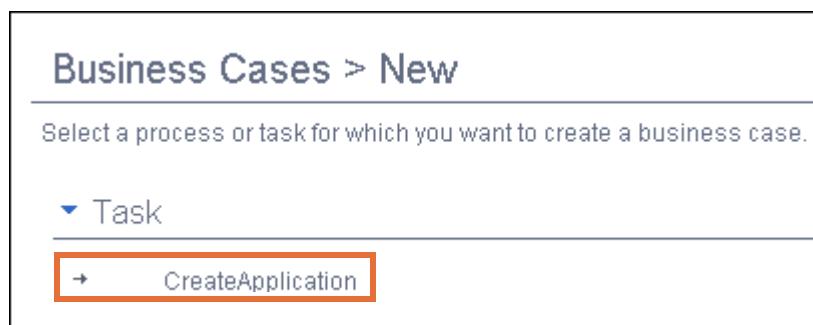
The screenshot shows the 'Login to Business User Client' page. At the top, it says 'Enter user name and password, then click Login.' Below that, there are two input fields: 'Name:' containing 'admin' and 'Password:' containing '*****'. A red box highlights the 'Login' button at the bottom.

- __ c. In the **Business Case** section of the Business User Client page, click **New**.



The screenshot shows the 'Business User Client' page with a 'Business Case' section. Under 'Business Case', there is a link labeled '+ New' which is highlighted with a red box. Below the link, a note says: 'Select this to view a list of all the tasks that you can use to create a business case. criteria, you can provide additional information.'

- __ d. In the **Task** section of the **Business Cases > New** page, click **CreateApplication**.



The screenshot shows the 'Business Cases > New' page. It has a note: 'Select a process or task for which you want to create a business case.' Below that, there is a 'Task' section with a link labeled '+ CreateApplication' which is highlighted with a red box.

- __ e. In the **Input Data** section of the **Business Cases > New > CreateApplication** page, enter TestCo in the **companyName** field. Leave the other fields blank and click **Create** at the bottom of the page.

Business Cases > New > CreateApplication

Enter the values for the input data and optionally provide additional information to create your task.

▼ Input Data

accountNumber	<input type="text"/>
applicationDate	<input type="text"/>
applicationDecision	<input type="checkbox"/>
comments	<input type="text"/>
companyName	<input type="text" value="TestCo"/>
contactFirstName	<input type="text"/>
contactLastName	<input type="text"/>

The **Business Cases > New** page opens again. The application processes the input data up to the Request More Documentation human task.

- __ f. Minimize the browser window.



Note

If you see an error in the Server Logs view similar to A WebGroup/Virtual Host to handle /favicon.ico has not been defined, it is safe to ignore it.

-
- __ 5. Using the Business Process Choreographer Explorer client, complete the Request More Documentation task.
- __ a. Switch to the **Servers** view in IBM Integration Designer.
 - __ b. Right-click **IBM Process Server V8.5.7 at localhost** and choose **Launch > Business Process Choreographer Explorer** from the menu.
 - __ c. If the security alert window prompts you, click **Yes** two times to proceed.
 - __ d. In the Business Process Choreographer Explorer login page, use **User Name** admin and **Password** web1sphere and click **Login**. The **My To-dos** page opens.

- __ e. A task is shown in the task list for you to claim. Click the **Request More Documentation** link.

<input type="checkbox"/> Priority ◊	Task Name ◊	State ◊	Kind ◊	Owner ◊
<input type="checkbox"/> 5	Request More Documentation	Ready	To-do Task	

The **Task Instance** page is opened.

- __ f. Click **Work on** to claim the task.

The screenshot shows the 'Task Instance' page. At the top, there is a header with the title 'Task Instance' and a sub-instruction: 'Use this page to display information about the task and, option...'. Below the header is a row of buttons: 'Work on' (which is highlighted with a red box), 'Terminate', 'Suspend', 'Restart', and 'Work Items'. Underneath these buttons is a section titled 'Task Description'. This section contains three rows of information: 'Task Name' (Request More Documentation), 'Description' (Request More Documentation sta...), and 'Reason' (Administrator, Originator).

The **Task Message** page opens.

- __ g. Scroll to the **Task Output Message** section, and click **Edit Source** at the bottom of the section.

The screenshot shows the 'Task Output Message' section. It displays a list of message names: 'pricingCode', 'pricingScore', 'productName', and 'requestAccountAmount'. At the bottom of the list is a button labeled 'Edit Source' (which is highlighted with a red box).

- __ h. Scroll to the bottom of the page, to the **Task Output Message Source View** section.
 __ i. In Windows Explorer, go to C:\labfiles\Support Files\EX3.
 __ j. Open EX3_Test_Data.txt in a text editor such as Notepad.

- ___ k. Copy the text from the file (use **Ctrl+A** to select all of the lines, and **Ctrl+C** to copy). Then, paste it (use **Ctrl+V**) over the existing text in the **Task Output Message Source View** window.

Task
Output
Message

Source View*

```
<?xml version="1.0" encoding="UTF-8"?>
<p:InputCriterionResponse xsi:type="p:InputCriterionResponse_._typ"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:p="http://FoundationLibrary/RequestMoreDocumentation">
  <Output>
    <accountNumber>TEST001</accountNumber>
    <applicationDate>July 17, 2014</applicationDate>
    <applicationDecision>false</applicationDecision>
    <comments>Complete</comments>
    <companyName>TestCo</companyName>
    <contactFirstName>Jane</contactFirstName>
    <contactLastName>Doe</contactLastName>
    <contactPhoneNumber>872-555-9915</contactPhoneNumber>
    <creditRating>C</creditRating>
    <creditReportNeeded>true</creditReportNeeded>
    <creditRisk>HIGH</creditRisk>
    <creditScore>1</creditScore>
    <customerCity>Chicago</customerCity>
    <customerCountry>USA</customerCountry>
```

- ___ l. Click **Confirm**. You are returned to the top of the Task Message page.
- ___ m. Scroll to the **Task Output Message** section again, and verify that the **Comments** field contains the value `Complete`.

Task
Output
Message

Form View*

Output	accountNumber	TEST001
	applicationDate	July 17, 2014
	applicationDecision	<input type="checkbox"/> Remove
	comments	Complete

- ___ n. Scroll to the top of the page and click **Complete** to finish the task.
- ___ o. Leave the Windows Explorer session open, but close the `EX3_Test_Data.txt` file.

- ___ 6. Set the value of the `applicationDecision` field to `false` to decline the application.
 - ___ a. In the Business Process Choreographer Explorer navigator pane, scroll to the **Task Instances** section and click **My To-dos**. The list of tasks gets updated. A **Final Application Review** task is shown.
 - ___ b. Click the check box to the left of the **Final Application Review** task, and click **Work on**.

Priority	Task Name	State	Kind
5	Final Application Review	Ready	To-do Task

- ___ c. Scroll to the bottom of the Task Message page, to the **Task Output Message** section, and click **Edit Source** at the bottom of the section.
- ___ d. In Windows Explorer, go to `C:\labfiles\Support Files\EX3`.
- ___ e. Open `EX3_Test_Data2.txt` in a text editor such as Notepad.
- ___ f. Copy the text from the file (use **Ctrl +A** to select all of the lines, and **Ctrl+C** to copy). Then, paste it (use **Ctrl+V**) over the existing text in the **Task Output Message Source View** window.
- ___ g. Click **Confirm**. You are returned to the Task Message page.
- ___ h. Scroll to the **Task Output Message** section and verify that **applicationDecision** is set to `false` (the box is not checked).

accountNumber	TEST001
applicationDate	July 17, 2014
applicationDecision	<input type="checkbox"/> Remove

- ___ i. At the top of the page, click **Complete**. The remainder of the AccountVerification flow runs, and the application is declined.

Part 4: Reviewing the test results

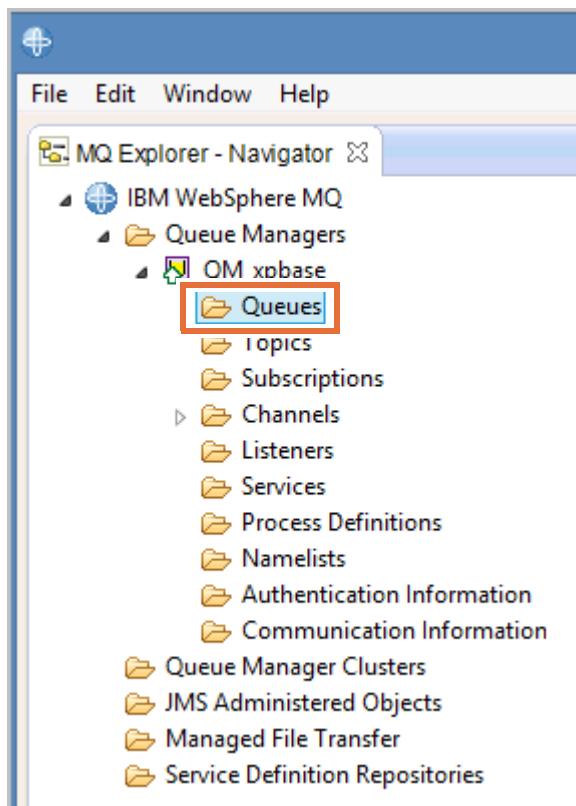
In this part of the exercise, you view the results of the test to verify that an “application declined” message is sent to the appropriate WebSphere MQ queue. You verify the contents of the `DeclinedApplication` queue by using both WebSphere MQ Explorer and the `RFHUtil` utility.

RFHUtil is a utility that can be used to browse WebSphere MQ queues and also the JMS queues. It is available for download as MQ SupportPac IH03. For more information about RFHUtil, browse the WebSphere MQ product offering web page at: www.ibm.com/software/wmq

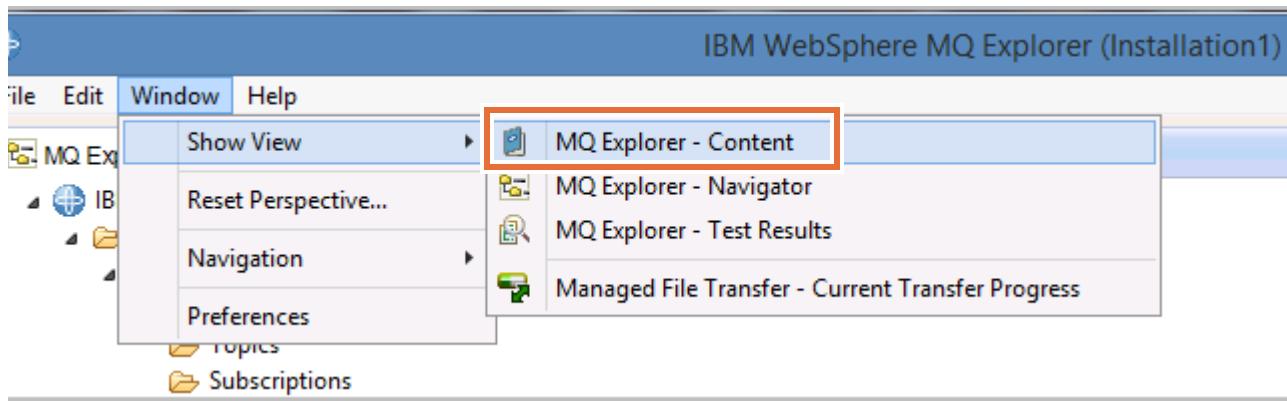
- 1. Verify the process execution path by using the Server Logs view in IBM Integration Designer.
 - a. Switch to the **Server Logs** view.
 - b. Review the log messages. The messages **Record Declined Application - begins** and **Record Declined Application - ends** indicate that the RecordDeclinedApplication activity was completed.

```
[Java] Determine Applicant Eligibility - begins
[Java] Determine Applicant Eligibility - ends
[Java] Generate Decline - begins
[Java] Generate Decline - Account for customer TestCo was declined and the credit risk was HIGH
[Java] Generate Decline - ends
[Java] Record Declined Application - begins
[Java] Record Declined Application - ends
```

- 2. Check for the application declined message in WebSphere MQ queue.
 - a. On the desktop, double-click the **WebSphere MQ Explorer** shortcut.
 - b. From the Navigator pane on the left, expand **IBM WebSphere MQ > Queue Managers > QM_xpbase**, and select **Queues**.



- ___ c. Go to **Window > Show view** and click **MQ Explorer - Content** to view the Queue detail pane on the right side of the screen.



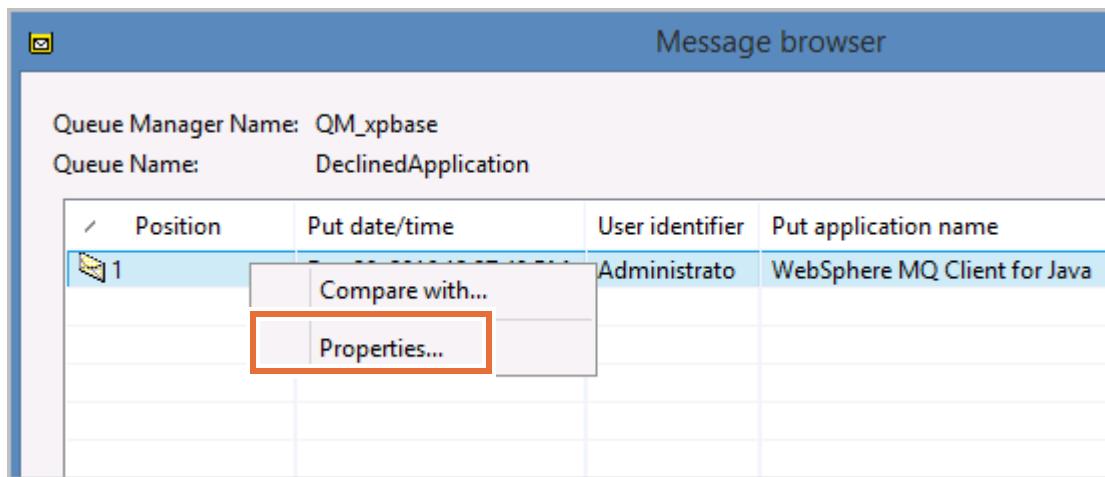
- ___ d. In the right pane, note the details of the queues that the QM_xpbase queue manager controls, specifically the DeclinedApplication queue. It shows a current queue depth of 1, which means a message is queued.

The screenshot shows the 'Queues' table in the 'MQ Explorer - Content' window. The table has columns for Queue name, Queue type, Open input count, Open output count, Current queue depth, and Put message allowed. One row is visible for the queue 'DeclinedApplication', which is highlighted with a red box. The queue type is 'Local', and its current queue depth is '1'. The 'Put message allowed' column shows 'Allowed'.

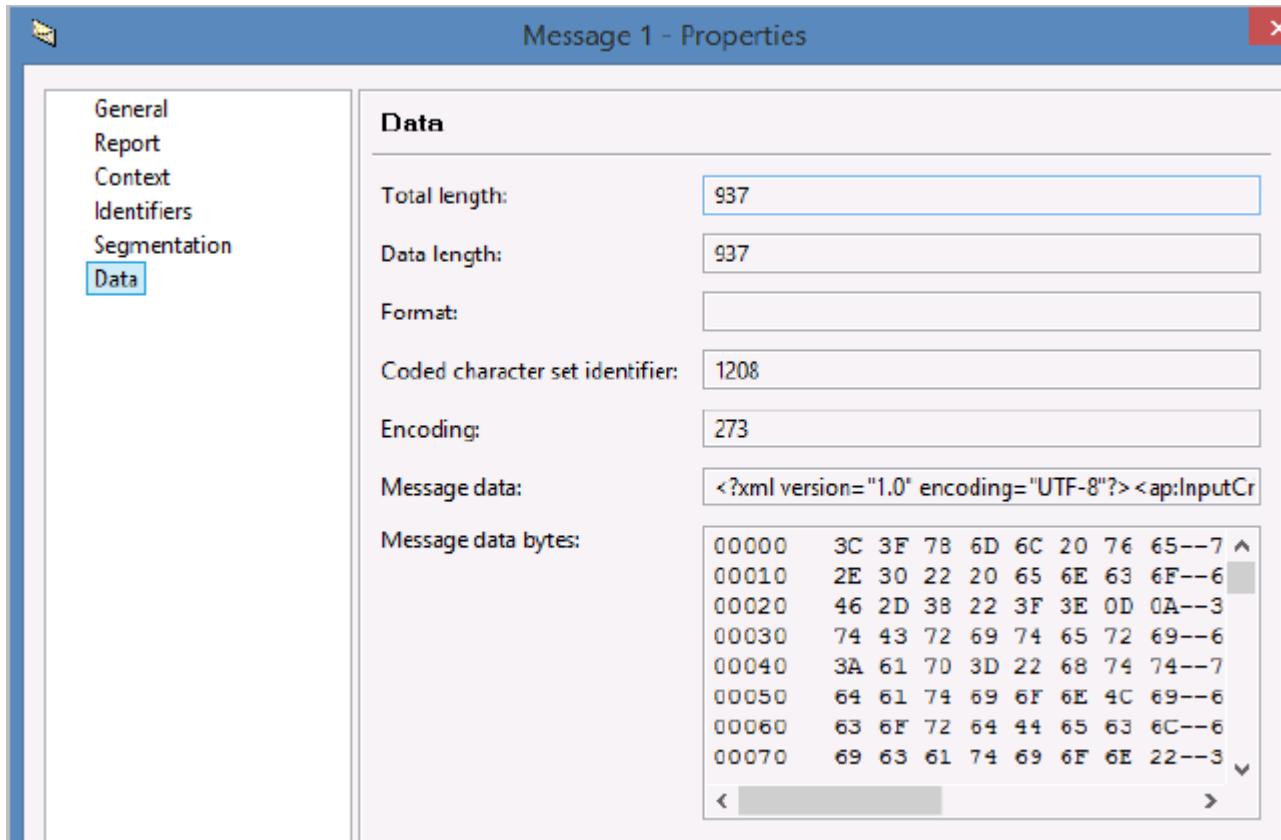
Queue name	Queue type	Open input count	Open output count	Current queue depth	Put message allowed
DeclinedApplication	Local	0	0	1	Allowed

- ___ e. To examine the details of the message, right-click the **DeclinedApplication** row of the queues table, and select **Browse Messages** from the menu. The IBM MQ message browser opens.
- ___ f. Scroll across the row to see the various types of control information that IBM MQ adds to manage the message while it is in transport. These types of control information might include the accounting token, the encoding information, and the expiry value.

- __ g. To view the data in detail, right-click the row of message data that is contained in the queue table and select **Properties**. The Properties window opens.

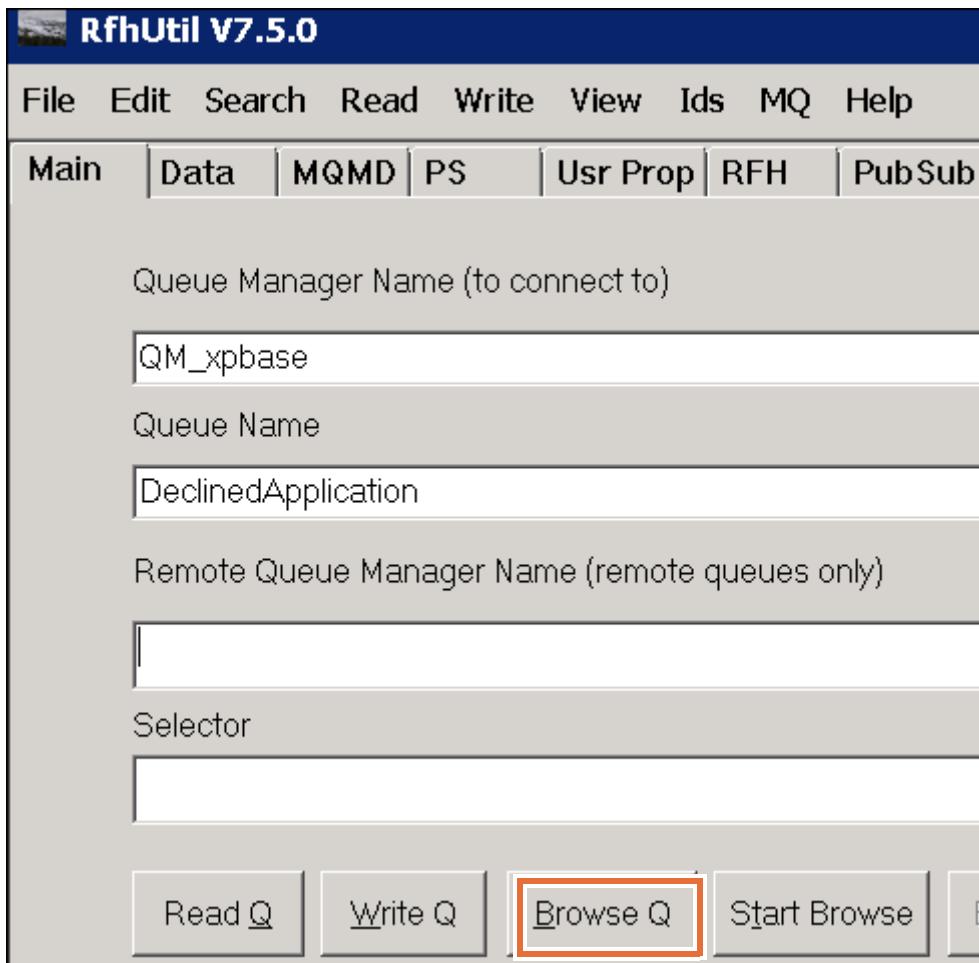


- __ h. In the navigator pane of the Properties window, click **Data**. The content of the message is shown in character and hexadecimal format.



- __ i. When you are done browsing, click **Close** in the Properties window and click **Close** in the **Message Browser** window.
- __ j. Close IBM MQ Explorer by choosing **File > Exit** from the menu options.

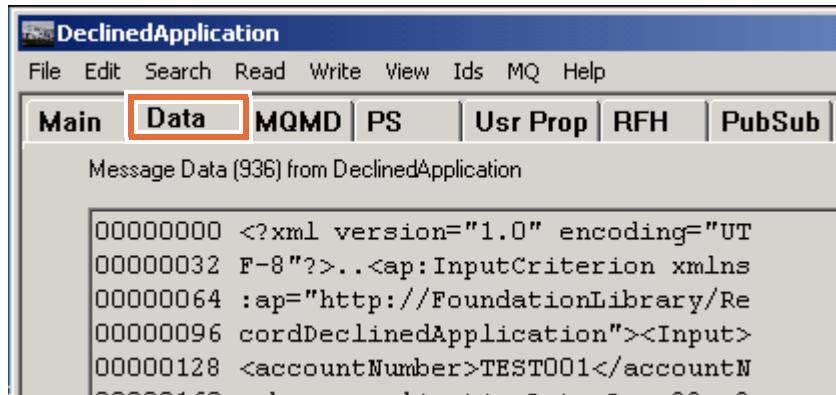
- __ 3. Use the RFHUtil program to browse the message in the **DeclinedApplication** queue.
 - __ a. Start RFHUtil by double-clicking `rfhutil.exe` in `C:\Tools\RFHUtil`. The main window of RFHUtil is opened.
 - __ b. If it is not already selected, click the **Main** tab
 - __ c. For the **Queue Manager Name** field, select **QM_xpbase** from the list.
 - __ d. For the **Queue Name** field, select **DeclinedApplication** from the list.
 - __ e. Click **Browse Q**.



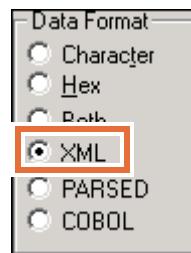
The message that shows in the window at the bottom of the pane contains the message length.

16.24.12 Msg browse from DeclinedApplication length=937

- ___ f. Click the **Data** tab.



- ___ g. For readability, in the **Data Format** section, select the **XML** option.



- ___ 4. (Optional) When you are done browsing the queue, you can delete the message by returning to the **Main** tab and selecting **Purge Q**. You can also delete messages from within WebSphere MQ Explorer.
- ___ 5. Close RFHUtil.
- ___ 6. Stop the WebSphere MQ service.
 - ___ a. Maximize the **Services** console that you minimized earlier and locate the **IBM WebSphere MQ** service in the Services window.
 - ___ b. Right-click **IBM WebSphere MQ** and choose **Stop** from the menu. The service is stopped when the **Status** column is blank.

Name	Description	Status
Hyper-V Volume Shadow Copy Re...	Coordinates the co...	
IBM Secure Shell Server for Windows	IBM Secure Shell Se...	Running
IBM WebSphere MQ (Installation1)	Provides startup an...	
IKE and AuthIP IPsec Keying Modu...	The IKEEXT service ...	Running
Interactive Services Detection	Enables user notific...	

- ___ c. Close the Services window by clicking **File > Exit**.

Part 5: Cleaning up the workspace

Now if your testing is completed, you must remove the applications from the server. Because every workspace you create shares an instance of IBM Process Server, applications that are deployed in

one workspace remain installed on the server when you switch to a new workspace. To avoid this situation, be sure to remove any installed applications before switching the workspaces.

- 1. To properly stop and uninstall the application, all process and human task instances must also be stopped. Delete the running instances of human tasks.
 - a. In the Business Process Choreographer Explorer, click **All Tasks** under Task Instances to examine the running instances of human tasks.
 - b. Click **CreateApplication**.
 - c. Click **Delete** to delete the instance.



Note

If more instances of CreateApplication are displayed, then select each instance and click **Delete**.

- d. Log out of Business Process Choreographer Explorer and close the tab.
- e. Close the browser window, close `EX3_Test_Data2.txt`, and close Windows Explorer.
- 2. Remove the deployed projects from the server.
 - a. In the **Servers** view, right-click **IBM Process Server v8.5.7 at localhost** and choose **Add and Remove** from the menu.
 - b. Click **Remove All**.
 - c. Click **Finish** and wait until the projects are removed from the server. It might take several minutes.
- 3. Close the Integration Designer workspace.

Using the event sequence quality of service qualifier

Event sequencing is one of the qualities of service (QoS) qualifiers that IBM Process Server supports. With event sequencing, you can define one or more “keys” that are used to group messages. When event sequencing is enabled, and two or more messages that have the same key value are received at an interface, those messages are prevented from being processed simultaneously.

It is important to note here that the event sequencing does **not** explicitly force a specific processing order that is based on the value of the sequencing keys. It also does not physically store or reorder messages that are based on sequencing key values. It prevents only messages that have the same key value as another from being processed at the same time.

For example, suppose that five messages are received, and their sequencing keys have the values 4, 5, 2, 4, 5. The first three messages (4, 5, 2) are eligible to be processed simultaneously, but the second “4” message cannot be processed until the first “4” message is processed. Similarly, the second “5” message cannot be processed at the same time as the first “5” message; it can be processed only after the first “5” message processing is complete.

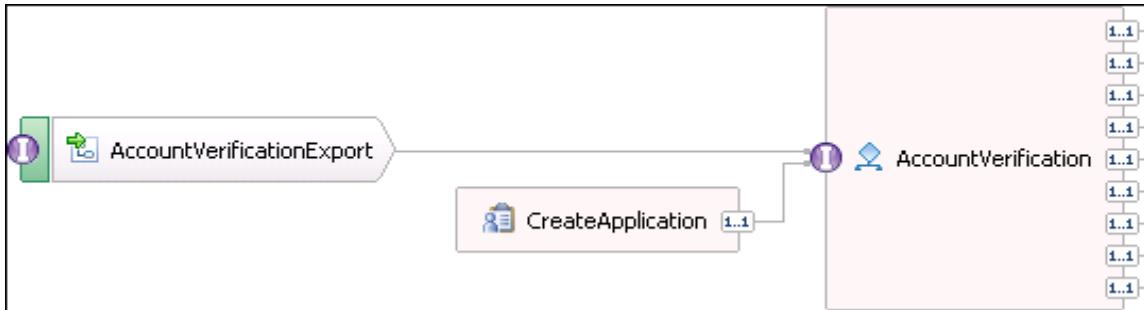
Event sequencing can be enabled only for asynchronous calls, and only on interfaces. In this exercise, you enable event sequencing on a flat file adapter.

Part 1: Adding the flat file adapter

The workspace already contains a module that you use to implement the event sequencing qualifier. First, you add the flat file adapter to the module. You can process multiple input records from a file, rather than through the existing web interface. Processing several records in a single run, you can see how event sequencing operates.

- ___ 1. Open the Exercise 3b workspace.
 - ___ a. On your desktop, open the folder that is labeled **Exercise Shortcuts**.
 - ___ b. Double-click the shortcut that is labeled **Exercise 3b**. IBM Integration Designer starts.
 - ___ c. Close the **Getting Started** tab.
- ___ 2. Examine the assembly diagram for **FoundationModule**.
 - ___ a. In the **Project** section of the **Business Integration** pane, expand **FoundationModule**.

- __ b. Double-click **Assembly Diagram**. The assembly diagram opens in the editor pane.

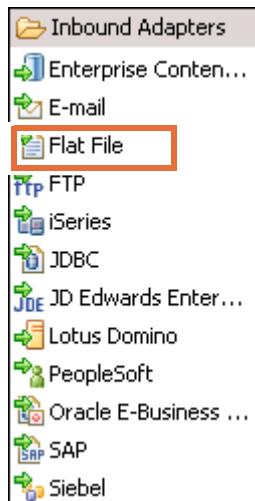


CreateApplication is the human task that provides input to the AccountVerification component. Normally, the CreateApplication user interface is used to test the application. The flat file adapter that you add takes the place of this human task.

- __ 3. Add a flat file adapter.

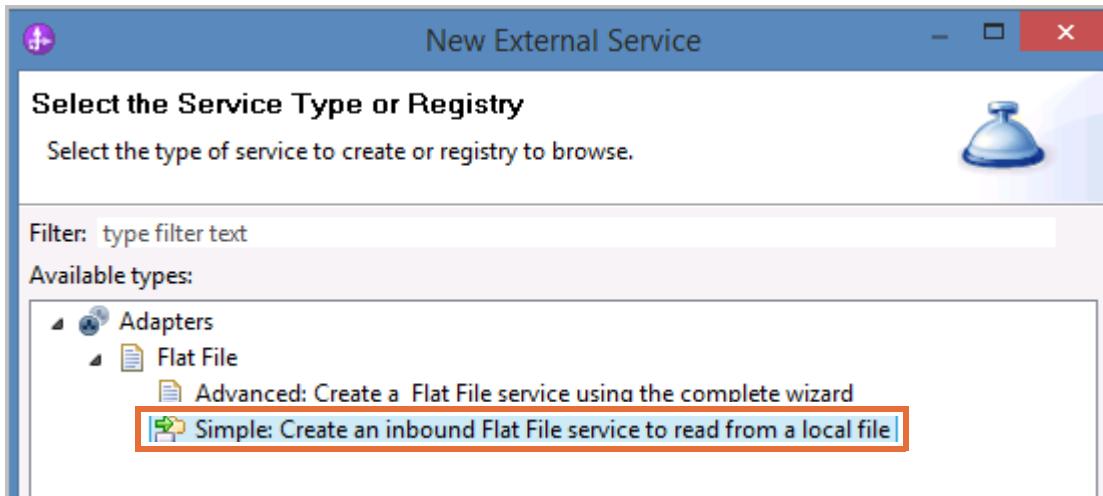
The WebSphere Adapter for Flat Files allows an application to read data from or write data to a flat file on the local file system. It is one of the WebSphere Technology Adapters that you can use to quickly build integration solutions without the need for writing code to manage the input and output operations. As mentioned previously, the flat file adapter is being used in this exercise to replace a human task. The adapter reads a file that contains test data. By testing with and without enabling the event sequencing, you are able to see the effect that event sequencing has on the integration solution.

- __ a. In the palette, expand the **Inbound Adapters** drawer.
__ b. Click the **Flat File** adapter.



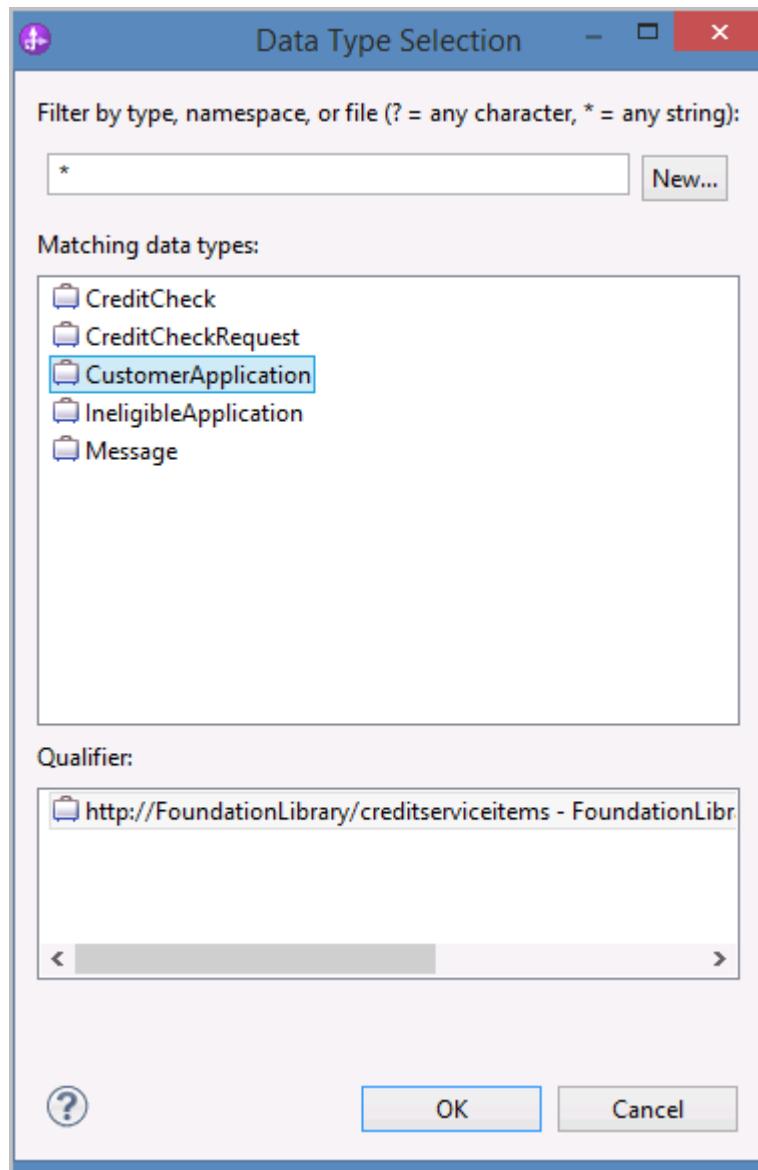
- __ c. In the drawing canvas, click below the **CreateApplication** human task icon. The **New External Service** window opens.

- __ d. Go to **Adapters > Flat File** and click the **Simple** wizard.



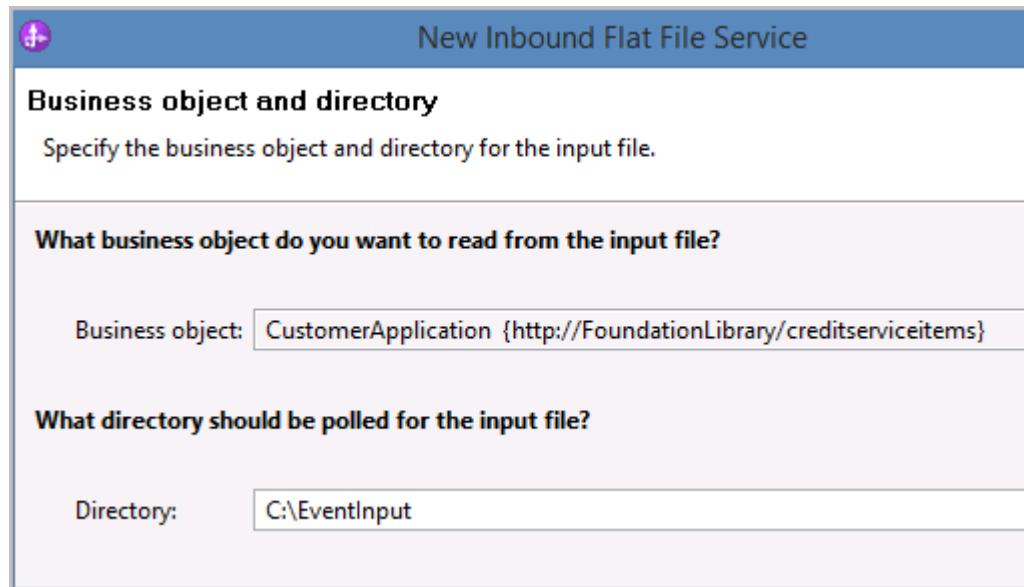
- __ e. Click **Next**. The **Flat File Service Name** window opens.
__ f. Click **Next** to accept the defaults. The **Business Object and Directory** window opens.
__ g. At the right of the **Business Object** field, click **Browse**. The **Data Type Selection** window opens.

- __ h. Click **CustomerApplication**, and then click **OK**.

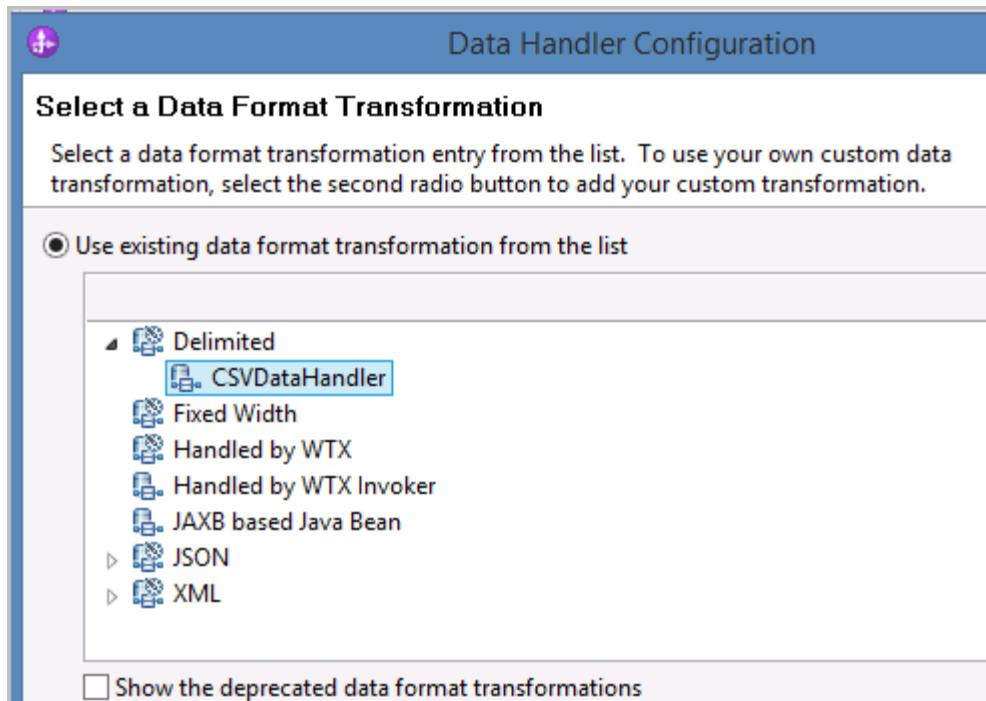


- __ i. At the right of the **Directory** field, click **Browse**.

- __ j. Go to C:\EventInput and then click **OK** (or you can directly enter the file path in the dialog box).



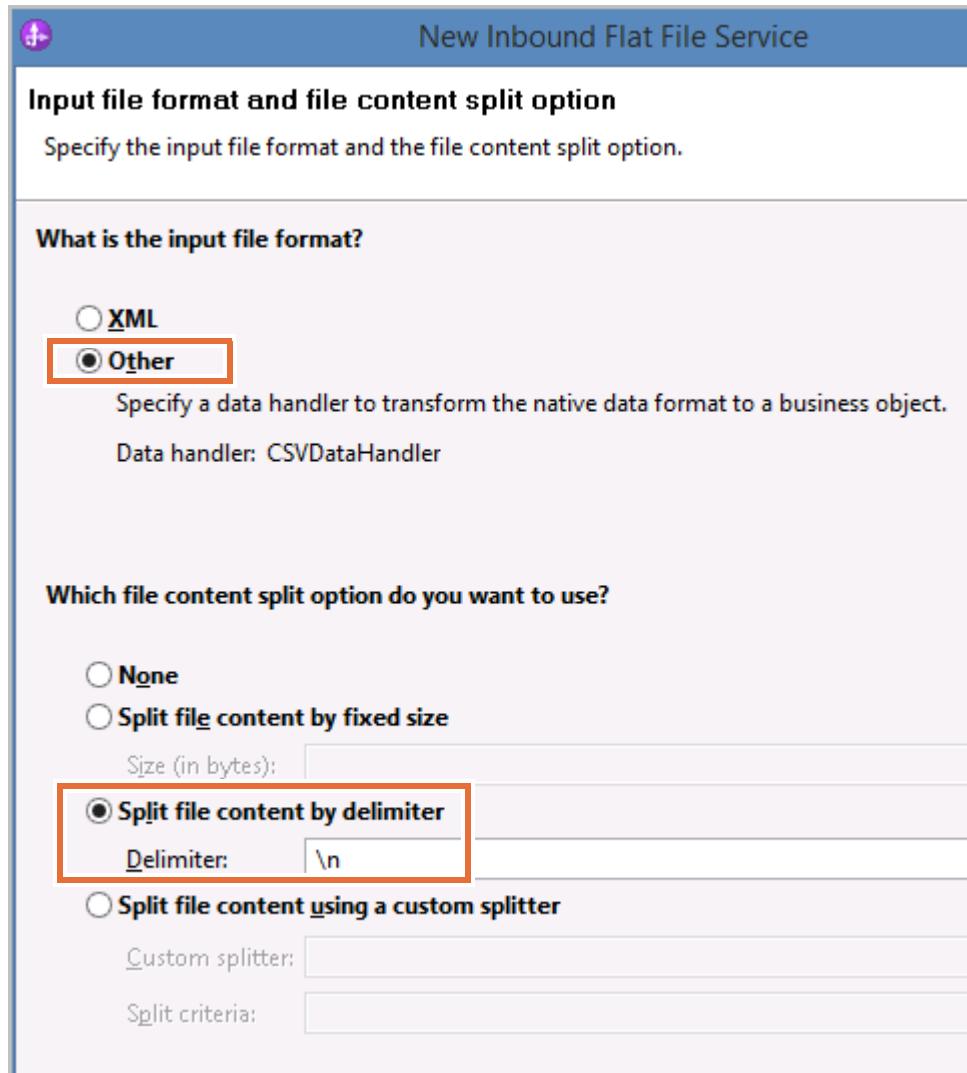
- __ k. Click **Next**. The **Input file format and file content split option** pane opens.
- __ l. Under the **What is the input file format** section, select **Other**.
- __ m. Click **Select** on the right side of the window. The **Select a Data Format Transformation** pane opens. Here, you specify how the input file is processed by specifying the characteristics of the input data.
- __ n. Expand **Delimited**, and then click **CSVDataHandler** to select the comma-separated values data handler.



- __ o. Click **Finish**.

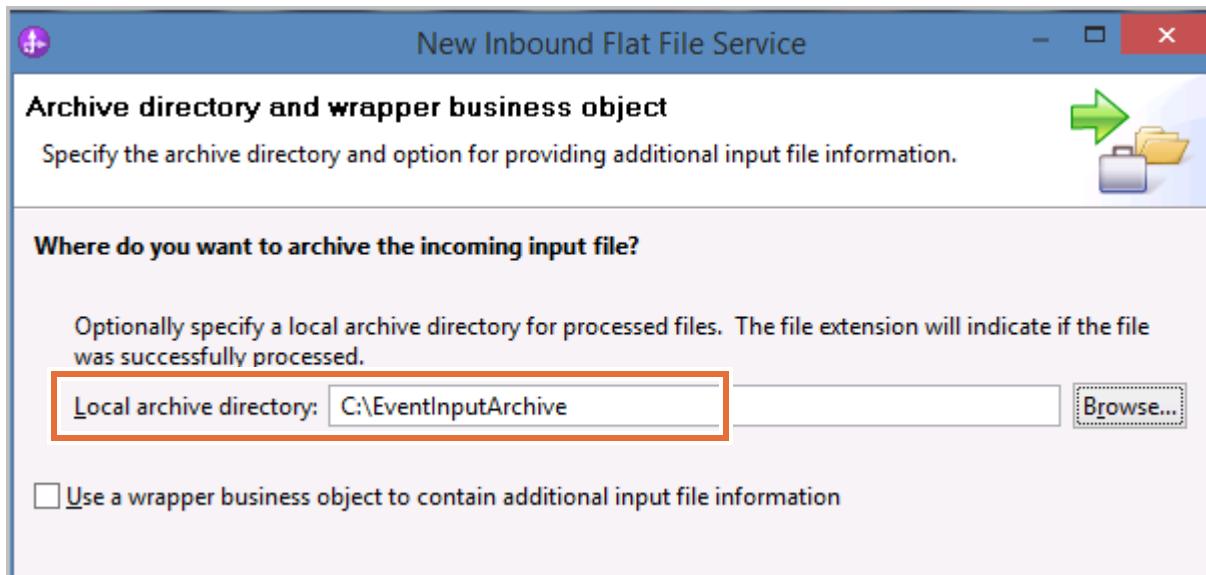
- ___ p. Under the **What file content split option do you want to use** section, select **Split file content by delimiter**.
- ___ q. In the **Delimiter** field, enter `\n` to specify the newline character.

The completed window now is as shown:

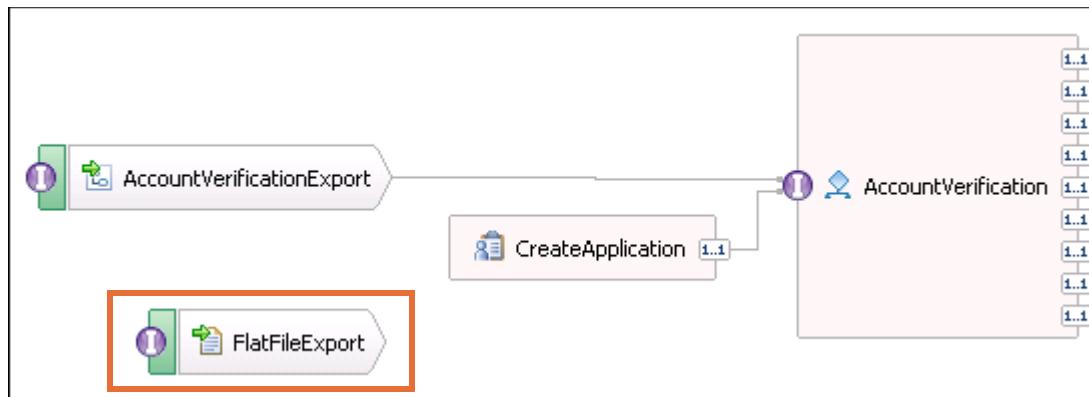


- ___ r. Click **Next**. The **Archive directory and wrapper business object** window opens.

- s. For the local archive directory, go to (or specify) C:\EventInputArchive. It is the location where the input files are moved after they are processed.



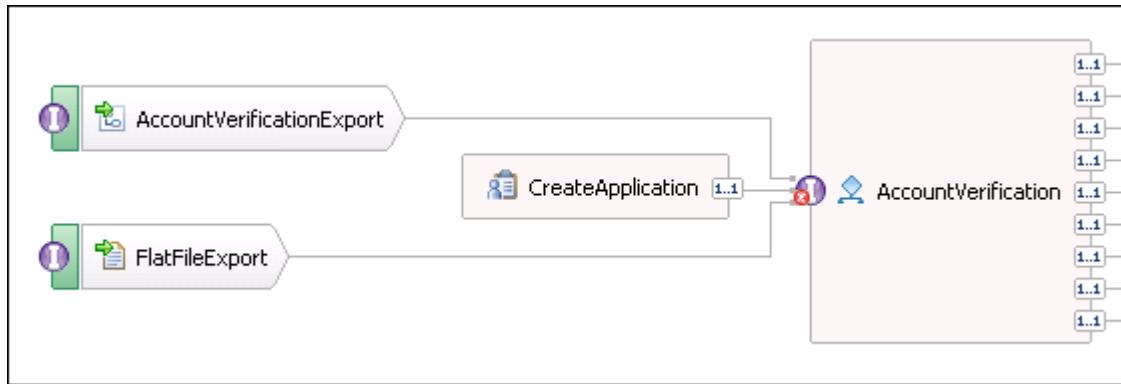
- t. Click **Finish**. The FlatFileExport is added to the canvas.



- 4. Wire the FlatFileExport to the AccountVerification process.

- a. Click the **FlatFileExport** component.

- ___ b. Hover the cursor over the right edge of the **FlatFileExport**. An orange wiring handle pops up at the right of the icon. Drag the handle to the interface of the **AccountVerification** component to wire them together. If an **Add Wire confirmation** window opens, click **OK**.

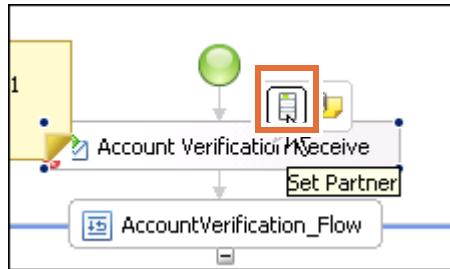


- ___ c. Save the workspace (**Ctrl+S**).
- ___ 5. Synchronize the new flat file interface to the implementation of the **AccountVerification** process. Because you just added the flat file interface, its implementation is not yet reflected in the component. Synchronizing the interface updates the component by including the implementation of the new interface.
 - ___ a. Right-click the **AccountVerification** component.
 - ___ b. From the menu, select **Synchronize Interfaces and References > To Implementation**.
 - ___ c. In the **Confirm Synchronize** window, click **Yes**. Wait for the workspace to rebuild. Any errors that are shown in the Problems view are resolved later in the exercise.
- ___ 6. Change the interface partner of the **AccountVerificationReceive** activity so that it receives messages from the **FlatFileExport**.
 - ___ a. Double-click the **AccountVerification** component to open it in the BPEL editor.
 - ___ b. Hover the cursor over the **Account Verification Receive** activity. The partner hover-over menu is opened. Notice that the process is associated with the **AccountVerification** interface partner.



- ___ c. Click the **Account Verification Receive** activity. An action bar is opened.

- ___ d. Click the **Set Partner** icon (the left icon).



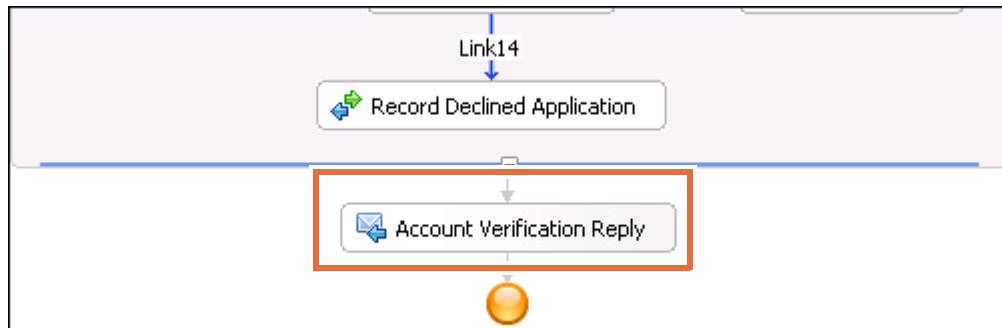
- ___ e. In the **Select Partner** window, click **FlatFileExport**, and then click **OK**.



- ___ f. Select the **Account Verification Receive** activity, and switch to the **Properties** view.
- ___ g. Select the **Details** tab. Notice that the partner is updated, but the variable to receive the input from the partner operation is not specified.
- ___ h. Click the **none** value in the **Store into Variable** field. A list of variables is shown.
- ___ i. Click **CustomerApplicationVariable**. The Store into Variable field is updated.
- ___ j. Save the workspace (**Ctrl+S**).

- 7. Remove the reply operation from the AccountVerification process.
 The current **AccountVerification** module sends a response message to the **CreateApplication** web interface. Because the flat file adapter supports only one-way processing (inbound in this case), the interface has a one-way operation. Since the interface operation does not expect a response, you must remove the **Account Verification Reply** activity.

- a. Right-click the **Account Verification Reply** activity.



- b. From the menu, select **Delete**.
- 8. Save your changes. The **Problems** tab can indicate warnings and informational messages, but no errors. The module is ready for testing.
- 9. Close the process editor, but leave the assembly diagram open.

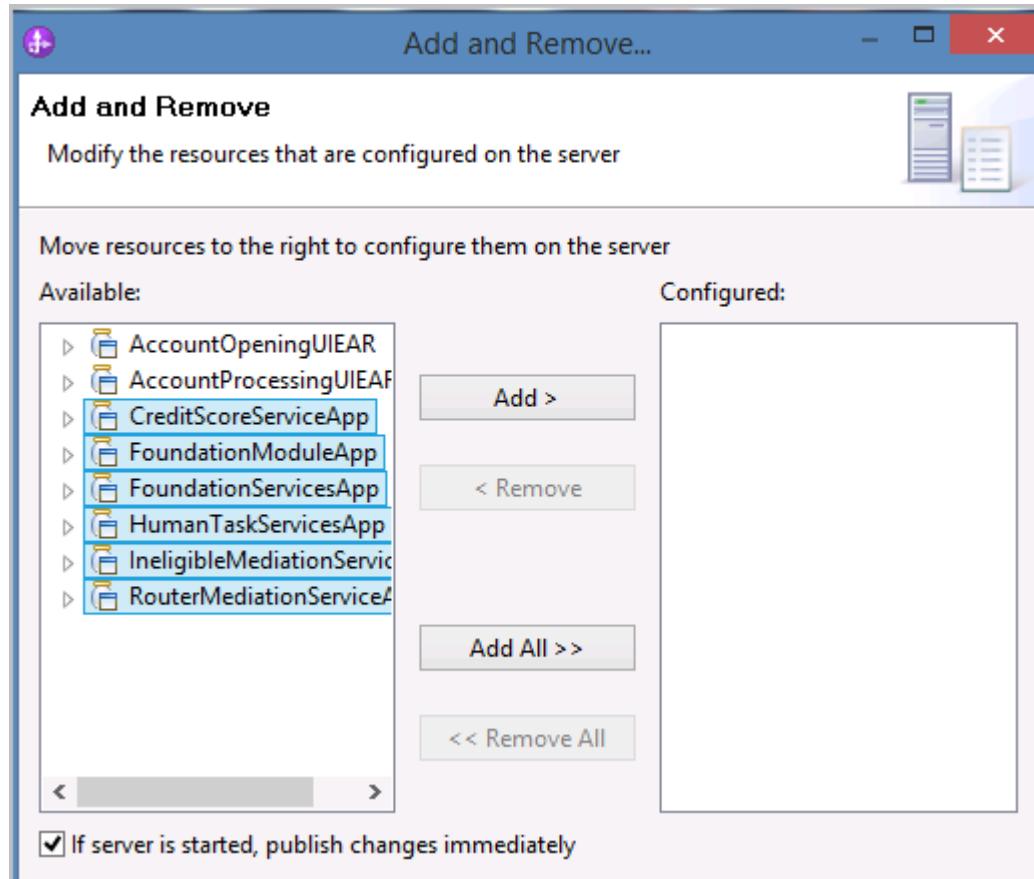
Part 2: Testing without event sequencing

You run two tests here, one without event sequencing, and a second with event sequencing enabled, so that you can see the effect that it has on the processing of incoming messages.

- 1. Start the server, unless it is already running.
 - a. In the **Servers** view, right-click the **IBM Process Server v8.5.7 at localhost** server.
 - b. From the menu, click **Start**.
 - c. Wait for the server to start before proceeding. The **State** column shows **Started**, and the server log view shows the message **Server server1 open for e-business**.
- 2. Deploy the application components.
 - a. In the **Servers** view, right-click **IBM Process Server v8.5.7 at localhost**.
 - b. Select **Add and Remove** from the menu.

- __ c. From the Available Projects window, select the following projects:

- CreditScoreServiceApp
- FoundationModuleApp
- FoundationServicesApp
- HumanTaskServicesApp
- IneligibleMediationServiceApp
- RouterMediationServiceApp



- __ d. Click **Add** to add them to the **Configured** list.
- __ e. Click **Finish**. The projects are deployed to the server. Wait for the deployment process to complete. The server **status** column shows **Publishing** while the deployment operation is occurring. It shows **Synchronized** when deployment is complete. It might take several minutes. Additionally, in the **Server Logs** view, you see that several messages are shown to indicate that the deployed applications are started successfully.



Note

It is not necessary to deploy the AccountOpeningUIEAR or AccountProcessingUIEAR projects. These projects are user interfaces that are not used to test the application.

- ___ 3. Test the application.
 - ___ a. Clear the server logs in the **Server Logs** view before running this test.
 - ___ b. Open Windows Explorer and go to `C:\labfiles\Support Files\Ex3`.
 - ___ c. Double-click the file that is named `Test_CustomerApplicationCSV.txt`. The file opens in a text editor.
 - ___ d. Review (but do not change) the contents of the test data. Notice that you see six records total: two records for AbcCo, then two records for IBM, then one record for AbcCo, and then the last record for IBM.
 - ___ e. Exit the editor.
 - ___ f. Right-click the file again, and select **Copy** from the menu.
 - ___ g. Go to `C:\EventInput` and paste the file into the directory.

The flat file adapter monitors the directory for files to process. When a file is shown in the directory, the adapter sends the records to the module for processing. After the processing is complete, the file is moved to the `C:\EventInputArchive` directory, where a suffix is added to the file name to indicate whether the file is successfully or unsuccessfully processed.

- ___ 4. Review the processing results in the **Server Logs** view.
 - ___ a. Switch to the **Server Logs** view.

- ___ b. Notice that the log record shows the company name.

Account verification failed for customer : AbcCo
Account verification failed for customer : 2014-08-01
>>> Invoking the Flat File Outbound service ...
<<< Flat File Outbound service invoked OK! ...
[Java] Record Ineligible Application - ends
[Java] Record Ineligible Application - begins
Account verification failed for customer : IBM
Account verification failed for customer : 2014-07-27
>>> Invoking the Flat File Outbound service ...
<<< Flat File Outbound service invoked OK! ...
[Java] Record Ineligible Application - ends
[Java] Record Ineligible Application - begins
Account verification failed for customer : AbcCo
Account verification failed for customer : 2014-07-31
>>> Invoking the Flat File Outbound service ...
<<< Flat File Outbound service invoked OK! ...
[Java] Record Ineligible Application - ends
[Java] Determine Applicant Eligibility - begins
[Java] Determine Applicant Eligibility - ends
[Java] Record Ineligible Application - begins
Account verification failed for customer : AbcCo
Account verification failed for customer : 2014-07-28
>>> Invoking the Flat File Outbound service ...
<<< Flat File Outbound service invoked OK! ...
[Java] Record Ineligible Application - ends
[Java] Record Ineligible Application - begins
Account verification failed for customer : IBM
Account verification failed for customer : 2014-07-29
>>> Invoking the Flat File Outbound service ...
<<< Flat File Outbound service invoked OK! ...
[Java] Record Ineligible Application - ends

- ___ c. Open the `Test_CustomerApplicationCSV.txt` file in an editor such as Notepad.

The six pairs of company name and application date values correspond to the company name and application date that are contained in the input file records.

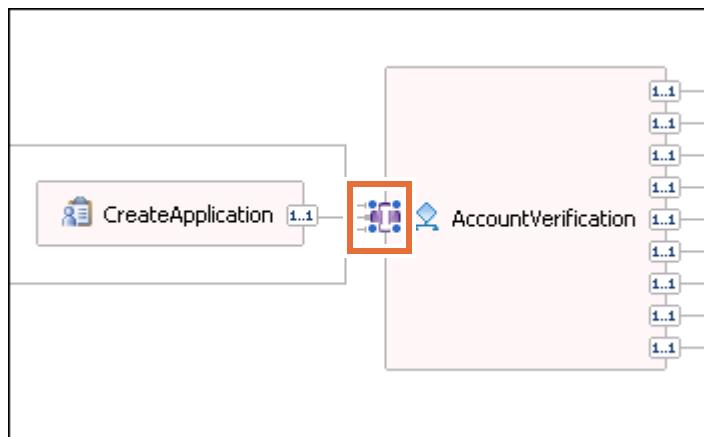
Test_CustomerApplicationCSV.txt - Notepad						
File	Edit	Format	View	Help		
ABC001,2014-08-01	true,None,AbcCc,John,Doe,547-555-1234					
ABC001,2014-07-31	true,None,AbcCc,John,Doe,547-555-1234					
IBM014,2014-07-30	true,None,IBM,Jane,Doe,914-376-0000					
IBM014,2014-07-29	true,None,IBM,Jane,Doe,914-376-0000					
ABC001,2014-07-28	true,None,AbcCc,John,Doe,547-555-1234					
IBM014,2014-07-27	true,None,IBM,Jane,Doe,914-376-0000					

- ___ d. Compare the data in the file with the log messages.

Part 3: Enabling the event sequencing qualifier

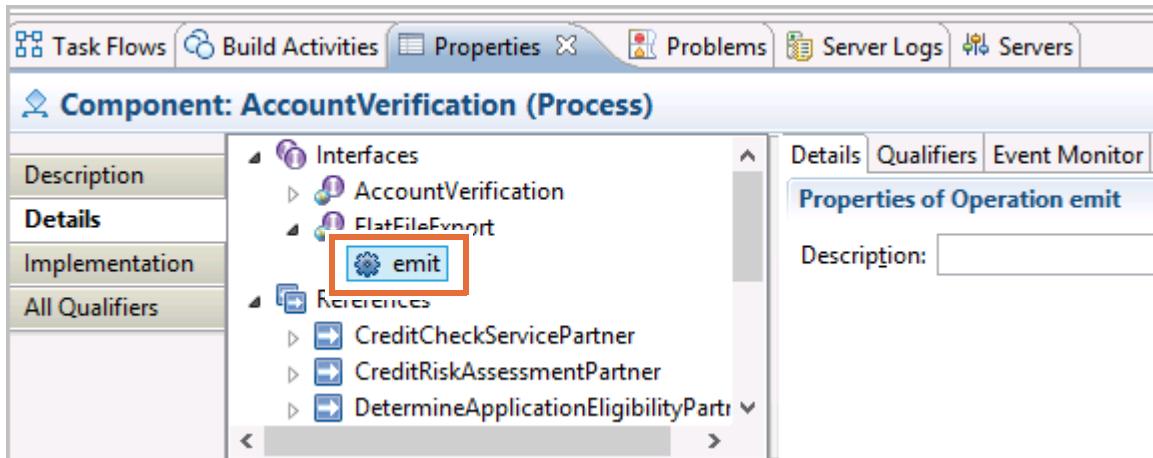
Now you add the event sequencing quality of service qualifier, which causes the processing of the input records in the text file. These records are processed in the same order that the event sequence key defines, rather than the order in which they appear in the file.

- ___ 1. Configure the event sequencing qualifier on the interface of the AccountVerification process SCA component.
 - ___ a. Switch to the assembly diagram for the FoundationModule.
 - ___ b. Select the interface on the AccountVerification component.

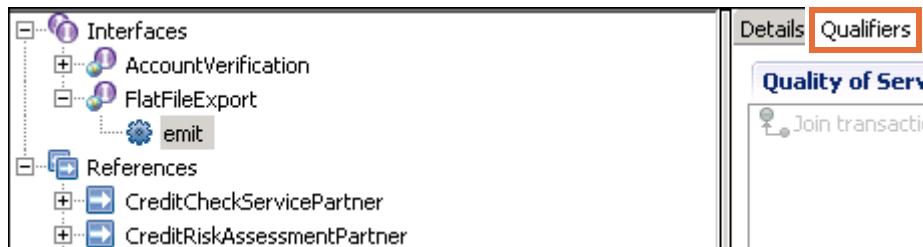


- ___ c. Switch to the **Properties** tab.
- ___ d. Select the **Details** tab.

- __ e. Expand **Interfaces > FlatFileExport**, and select the **emit** operation.

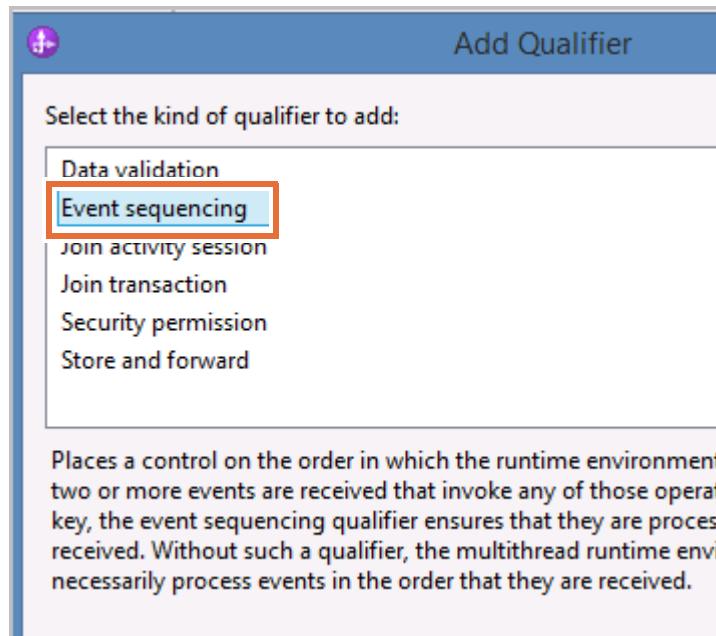


- __ f. In the properties pane to the right, select the **Qualifiers** tab.



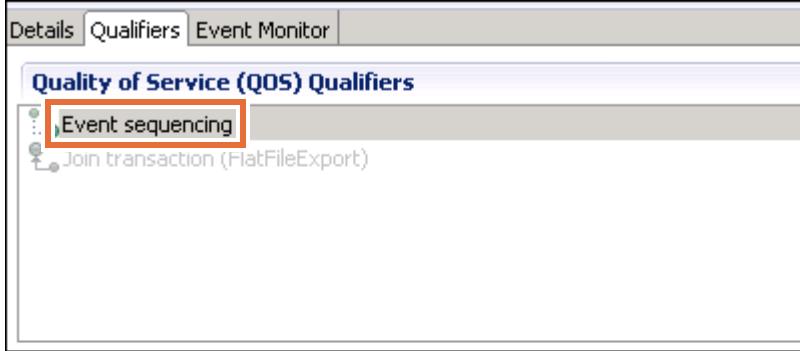
- __ g. Click **Add**. The **Add Qualifier** window is opened.

- __ h. Click **Event sequencing**.



- __ i. Click **OK**. The qualifier is added.

- __ j. Click **Event sequencing** in the **QoS Qualifiers** list. The properties for the qualifier are shown in the lower pane.



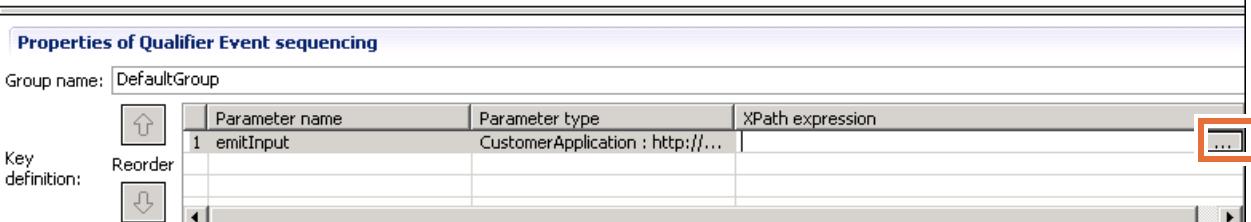
Quality of Service (QoS) Qualifiers

- Event sequencing
- Join transaction (FlatFileExport)

Properties of Qualifier Event sequencing

Group name:	DefaultGroup		
Key definition:	Reorder	Parameter name	Parameter type
	[Up] [Down]	1 emitInput	CustomerApplication

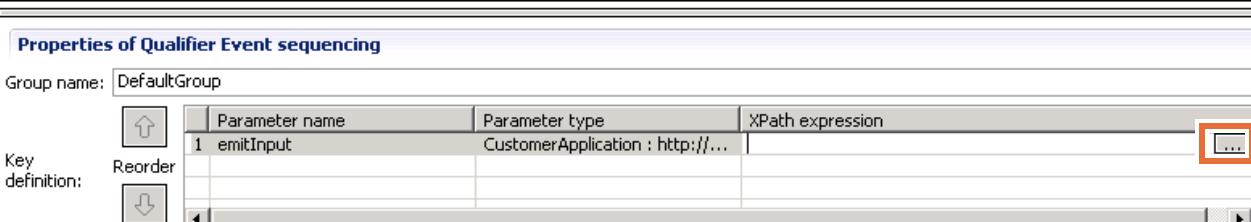
- __ 2. Specify the fields in the input data; it is used to control event sequencing.
- __ a. In the **Properties** view in the lower pane, click the **XPath expression** field in the first row of the table.



Properties of Qualifier Event sequencing

Group name:	DefaultGroup			
Key definition:	Reorder	Parameter name	Parameter type	XPath expression
	[Up] [Down]	1 emitInput	CustomerApplication : http://...	[...]

- __ b. Click the small box that contains the ellipsis at the right of the field.

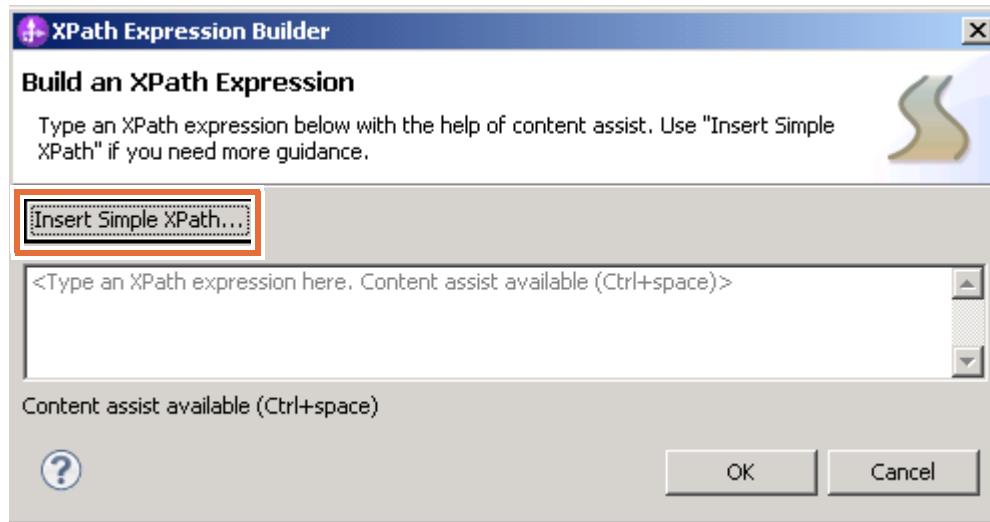


Properties of Qualifier Event sequencing

Group name:	DefaultGroup			
Key definition:	Reorder	Parameter name	Parameter type	XPath expression
	[Up] [Down]	1 emitInput	CustomerApplication : http://...	[...]

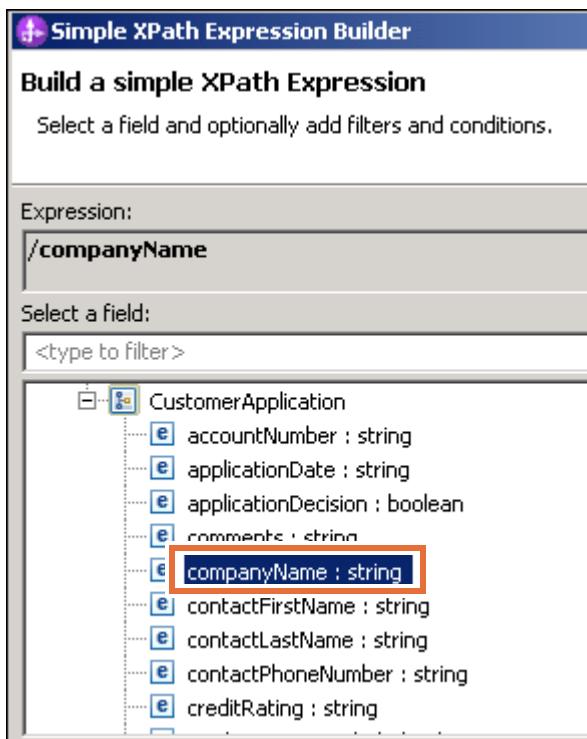
This action invokes the XPath expression builder to help you specify the XPath expression that is used as the event sequencing key.

- __ c. In the XPath Expression Builder window, click **Insert Simple XPath**.



The XPath expression builder window opens.

- __ d. In **Select a Field** section, expand **Data Types > CustomerApplication**.
__ e. Click **companyName**.



- __ f. Click **OK**. The XPath Expression Builder window is populated.



- __ g. Click **OK**. The XPath expression field in the sequencing key definition is populated.

Key definition:	Parameter name	Parameter type	XPath expression
<input type="button" value="Reorder"/>	1 emitInput	CustomerApplication : http://...	/companyName



Note

You can enter the `/companyName` value directly into the XPath expression field instead of using the XPath expression builder.

- __ h. Save all the workspace components (click **File > Save All** or **Ctrl+S**).

Part 4: Testing with event sequencing enabled

In this section, you test the application with event sequencing enabled on the interface and observe how records are processed.

- __ 1. Redeploy the updated application components. One way to do that is to publish the projects again to the server. However, in this exercise you remove all the projects from the server first and then add them back. Otherwise, you might get an exception during the test.
 - __ a. In the **Servers** view, right-click **IBM Process Server v8.5.7 at localhost** and select **Add and Remove** from the menu.
 - __ b. Click **Remove All** and click **Finish**.

- ___ c. From the Available Projects window, select the following projects: **CreditScoreServiceApp**, **FoundationModuleApp**, **FoundationServicesApp**, **HumanTaskServicesApp**, **IneligibleMediationServiceApp**, and **RouterMediationServiceApp**. Click **Add** to add them to the **Configured** list.
- ___ d. Click **Finish**.
- ___ 2. Test the application.
 - ___ a. Clear the server logs in the **Server Logs** view before running this test.
 - ___ b. Open Windows Explorer by clicking the file folder in the taskbar, or right-click the Windows logo at the lower left and select **File Explore** from the menu.
 - ___ c. Go to `C:\labfiles\Support_Files\EX3`.
 - ___ d. Right-click the file that is named `Test_CustomerApplicationCSV.txt` and then select **Copy** from the menu.
 - ___ e. Go to `C:\EventInput` and paste the file into the directory.
- ___ 3. Review the processing results in the Server Logs view.
 - ___ a. Switch to the Server Logs view.
 - ___ b. Notice that the log record shows the company name and application date.

Contents
<code>Account verification failed for customer : IBM</code>
<code>Account verification failed for customer : 2014-07-30</code>
<code>Account verification failed for customer : AbcCo</code>
<code>Account verification failed for customer : 2014-07-31</code>
<code>Account verification failed for customer : IBM</code>
<code>Account verification failed for customer : 2014-07-29</code>
<code>Account verification failed for customer : AbcCo</code>
<code>Account verification failed for customer : 2014-08-01</code>
<code>Account verification failed for customer : AbcCo</code>
<code>Account verification failed for customer : 2014-07-28</code>
<code>Account verification failed for customer : IBM</code>
<code>Account verification failed for customer : 2014-07-27</code>



Important

If you receive an exception message or errors in the server logs and no new text file is created under `C:\EventInputArchive`, then empty the three folders `C:\EventInputArchive`, `C:\IneligibleAppArchive\staging`, and `C:\IneligibleAppArchive\outdir`. Then, run the test again by copying the text file one more time.

It is possible that your results are different. The reason is explained in a subsequent step.

- ___ c. Compare the data in the file (open the file with the text editor again if necessary) with the messages in the server log. Notice that the data from the file is likely not processed in the order the records are in the file, and the records are not likely processed in the same order as the previous test.

For this reason, the application can process records in parallel, which means that it is possible that more than one record processes at the same time without enabling the event sequencing. For example, two records for AbcCo can be processed at the same time, or two records for IBM, or a record for AbcCo and a record for IBM.

When event sequencing is enabled, it prevents two records that contain the same event sequence key value from being processed simultaneously. In this exercise, you defined the event sequence key to be the **companyName** field, which means that records with the same companyName value are going to process in sequential order, not in parallel. Thus, if the AbcCo records are processing, and another record becomes eligible for processing (in this case, read from the input file), its processing can begin if it is not another AbcCo record.

It is important to note that event sequencing does not imply that all records for company AbcCo are processed before all records for IBM. This idea is a common misconception.

- ___ d. Paste the `Test_CustomerApplicationCSV.txt` file into the `C:\EventInput` directory again and review the server log again. Repeat this one or more times to demonstrate that the order in which the input records are processed is indeterminate, but that no two records for a company are processed simultaneously. For this reason, the order of the records that shows in your server log might not match what is listed in the exercise instructions.

Even if event sequencing is not enabled, processing order is not deterministic. For this reason, it is possible that the record order in the first test might also not match the order that is listed in the exercise instructions.

Part 5: Cleaning up the workspace

- ___ 1. Remove the deployed projects from the server.
 - ___ a. In the **Servers** view, right-click **IBM Process Server v8.5.7 at localhost**.
 - ___ b. Select **Add and Remove** from the menu.
 - ___ c. Click **Remove All**.
 - ___ d. Click **Finish**. The projects are removed to the server. Wait for the deployment process to complete. The server **status** column shows **Publishing** while the deployment operation is occurring. It shows **[Started, Synchronized]** when deployment is complete. It might take several minutes.
- ___ 2. Stop the process server by selecting **IBM Process Server v8.5.7 at localhost** and clicking the **Stop the server** icon. It is important to stop the server for the next exercise.
- ___ 3. Close the IBM Integration Designer. Click **File > Exit**, or click **X** in the upper-right corner.

End of exercise

Exercise 4. Applying fault handlers

Estimated time

01:30

Overview

In business processes, some exceptions can be predicted and handled by defining fault handlers within the BPEL. Fault handlers can also be defined to handle unexpected, arbitrary faults.

Objectives

After completing this exercise, you should be able to:

- Implement a fault handler in a BPEL process
- Test fault handlers in the IBM Integration Designer integrated test environment

Exercise instructions

Preface

The following business objects are used in this exercise.

- **CustomerApplication:** The simple (non-hierarchical) business object that describes the characteristics of a particular customer application record. This generic business object (GBO) is instantiated from the user input through the human task.

The following interfaces are used along with the **CustomerApplication** business object.

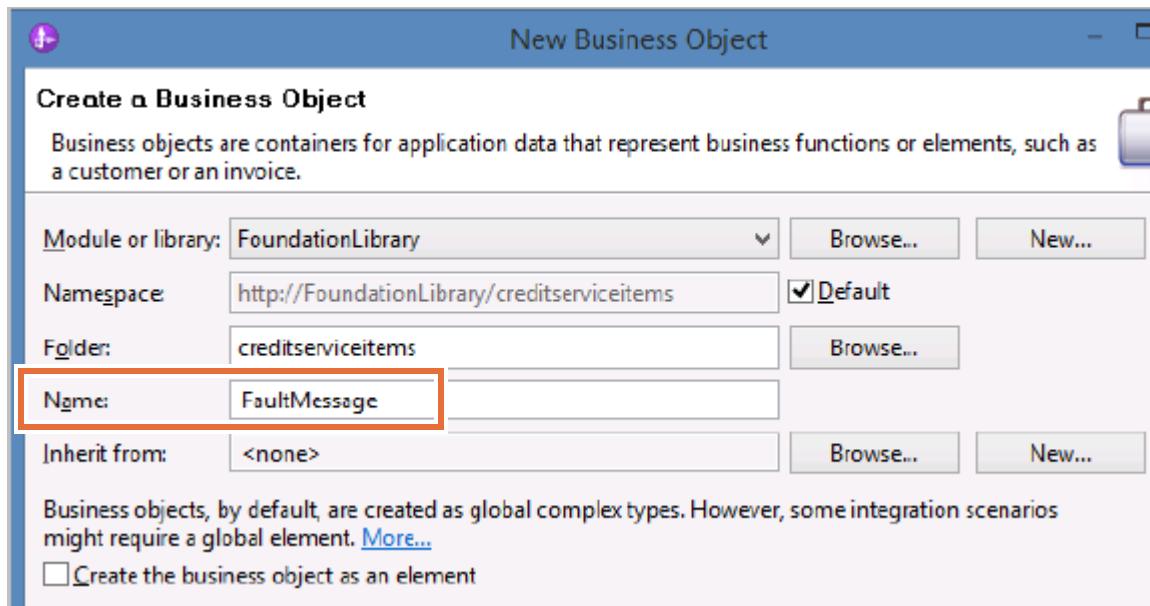
- **CustomerApplicationDelivery:** Contains a single one-way operation that passes a single part, named **inputCustomerApplication** of type **CustomerApplication**, on the request. This interface is used on the definition of the business process and exposed as part of the exported SCA binding, which the other SCA components call.
- **CustomerApplicationRequest:** Contains a single request/response operation that passes a single part, named **inputCustomerApplication** of type **CustomerApplication**, on the request; and a single part, named **outputCustomerApplication** of type **CustomerApplication**, on the response. With the use of this interface, it is expected that any changes that are made to **CustomerApplication** on the receiving end are passed back to the process. This interface is used to define reference partners in the business process. Using the same interface simplifies the data exchange between the processes.

Fault handling

Part 1: Creating a **FaultMessage** business object definition

In this lab exercise, you modify **FoundationLibrary** by defining a new data type (business object) definition. You build a new **FaultMessage** data type to encapsulate two string attributes, **FaultID** and **FaultData**, which are used to provide more information to the **AccountVerification** business process that catches the returned faults.

- ___ 1. Open the Exercise 4 workspace.
 - ___ a. On your desktop, open the folder that is labeled **Exercise Shortcuts**.
 - ___ b. Double-click the shortcut that is labeled **Exercise 4** to start IBM Integration Designer.
 - ___ c. Wait for the workspace build to complete; it might take a few minutes.
 - ___ d. Close the **Getting Started** tab.
- ___ 2. Verify that the **IBM Process Server v8.5.7 at localhost** is stopped. You stopped it at the end of the previous exercise. If the server is still running, then stop it before continuing to the next step, or you might get errors during testing. You start the process server later in this exercise.
- ___ 3. Create a data type (business object) definition in the **FoundationLibrary**.
 - ___ a. In the **Business Integration** view, expand **FoundationLibrary > Data**.
 - ___ b. Right-click **creditserviceitems** and then choose **New > Business Object** from the menu.
 - ___ c. At the **Business Object** screen, in the **Name** field, enter: **FaultMessage**

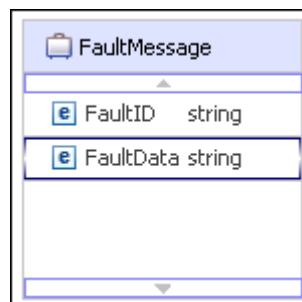


- ___ d. Accept the remaining default options and click **Finish** to create the business object. The business object editor tool opens automatically. The new **FaultMessage** business object definition can now also be found under **FoundationLibrary > Data > creditserviceitems**.

- ___ 4. Complete the implementation of the new **FaultMessage** business object definition.
 - ___ a. In the business object editor, click the **Add a field to a business object** icon.



- ___ b. Change the default attribute name to: `FaultID`
Leave the data type set to: `string`
- ___ c. Click the **Add a field to a business object** icon to add an attribute.
- ___ d. Change the default attribute name to: `FaultData`
Leave the type set to: `string`



- ___ e. Save your changes and close the business object editor.

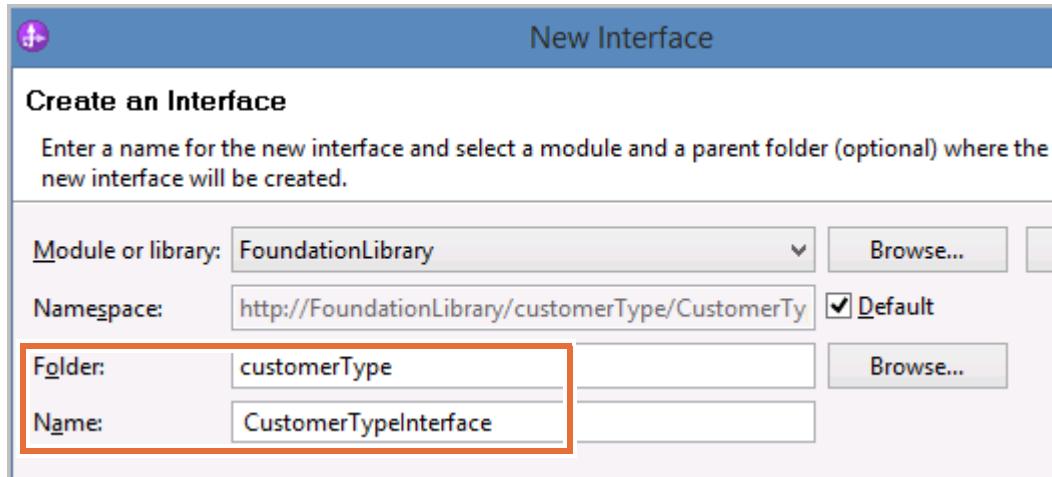
Part 2: Adding an interface definition in the FoundationLibrary

In this part of the lab, you modify **FoundationLibrary** by adding an interface definition. You build the **CustomerTypeInterface**, which defines a two-way request/response operation. This request/response operation passes a single part on the request, named **inputCustomerApplication** of type **CustomerApplication**, and a single part on the response, named **outputCustomerApplication** of type **CustomerApplication**. This interface also includes a fault response message, which you defined previously. With the use of this interface, any changes that are made to **CustomerApplication** on the receiving end are passed back to the process. If the target service fails to complete the requested operation, the fault message is returned.

- ___ 1. Add an **Interface** definition in the **FoundationLibrary**.
 - ___ a. In the **Business Integration** view, expand **FoundationLibrary > Interfaces**.
 - ___ b. Right-click **Interfaces** and choose **New > Interface** from the menu.

___ c. At the **Create a new interface** screen:

- In the **Folder** field, enter: `customerType`
- In the **Name** field, enter: `CustomerTypeInterface`



___ d. Click **Finish** to create the `CustomerTypeInterface`. The interface editor opens automatically. The new interface can now also be found under **FoundationLibrary > Interfaces** in the Business Integration view.

___ 2. Complete the implementation of the `CustomerTypeInterface`.

___ a. In the interface editor, beside the **Operations** section, click the **Add Request Response Operation** icon to define a two-way operation.



- ___ b. Change the default operation name to: `customerType`
- ___ c. Change the default input parameter name from `input1` to: `inputCustomerApplication`
- ___ d. For the input, click the default **string** type and select the **CustomerApplication** data type definition from the **Type** list.
- ___ e. Change the default output parameter name from `output1` to: `outputCustomerApplication`
- ___ f. For the output, click the default **string** type and select the **CustomerApplication** data type definition from the **Type** list.

- ___ g. Beside the **Operations** section, click the **Add Fault** icon to define the fault message parameter. If **Add Fault** is not selectable, click **customerType** and then **Add Fault**.

Operations and their parameters		
Name	Type	
customerType		
Input(s)	inputCustomerApplication	CustomerApplication
Output(s)	outputCustomerApplication	CustomerApplication
Fault	FaultMessage	FaultMessage

- ___ h. Change the default name to: `faultMessage`
 ___ i. For the fault, click the default **string** type and then select the **FaultMessage** data type definition from the **Type** list.

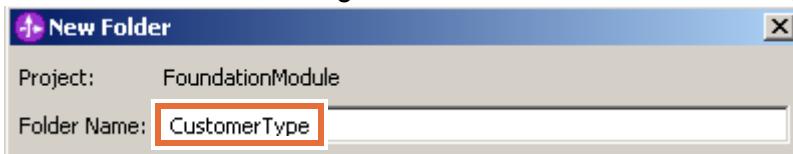
Operations and their parameters		
Name	Type	
customerType		
Input(s)	inputCustomerApplication	CustomerApplication
Output(s)	outputCustomerApplication	CustomerApplication
Fault	FaultMessage	FaultMessage

- ___ j. Save your changes and close the interface editor.

Part 3: Create a **CustomerType** BPEL business process

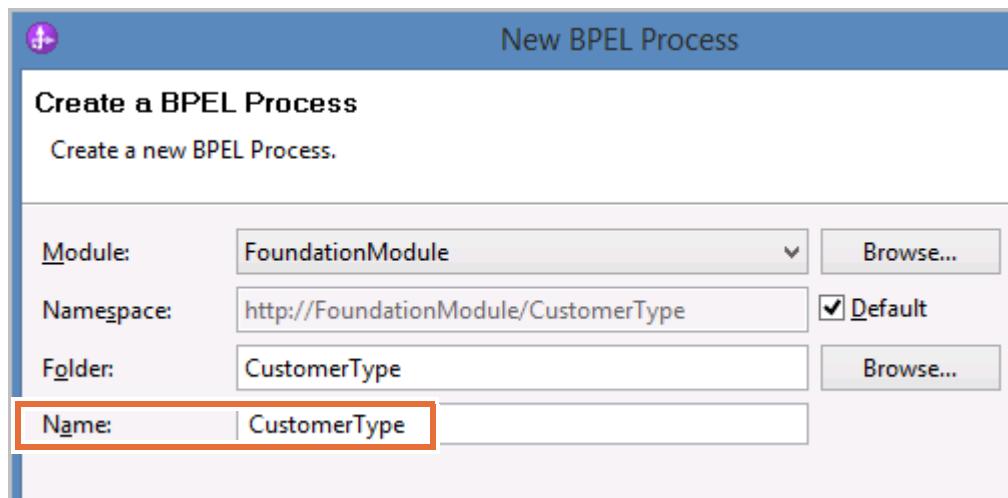
In this part of the lab, you modify **FoundationModule** by creating a BPEL process. You build a new **CustomerType** business process that uses the new **CustomerTypeInterface** and **FaultMessage** artifacts you defined.

- ___ 1. Create a **CustomerType** business process.
- In the **Projects** section of the **Business Integration** view, expand **FoundationModule** > **Integration Logic**.
 - Right-click **BPEL Processes** and choose **New > BPEL Process** from the menu.
 - At the **Create a BPEL Process** screen, for the **Folder** field, click **Browse**.
 - In the **Folder Selection** window, click **New Folder**.
 - In the **New Folder** window, change the **Folder Name** to: `CustomerType`

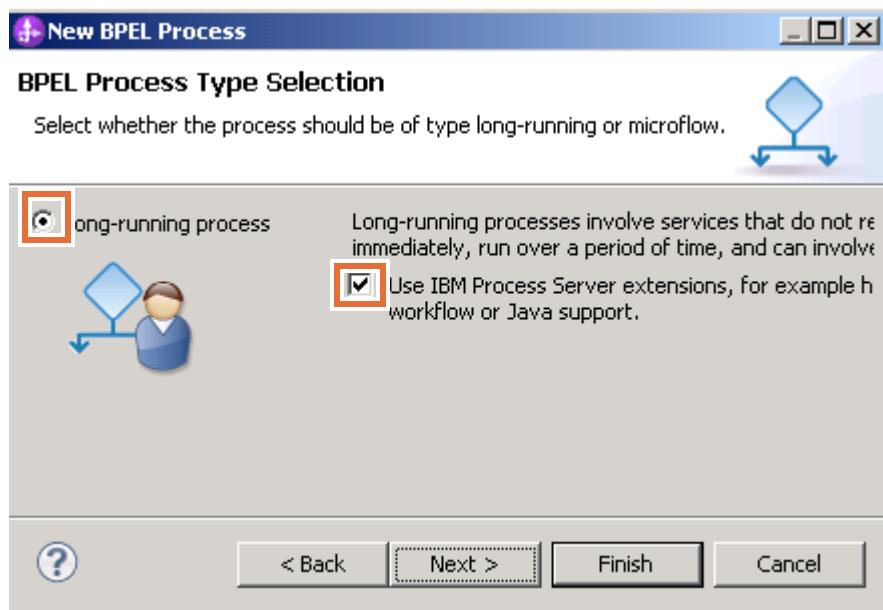


- ___ f. Click **OK**.
 ___ g. When you return to the **Folder Selection** window, click **OK**.

- ___ h. When you return to **Create a Business Process** screen, in the **Name** field, enter:
CustomerType

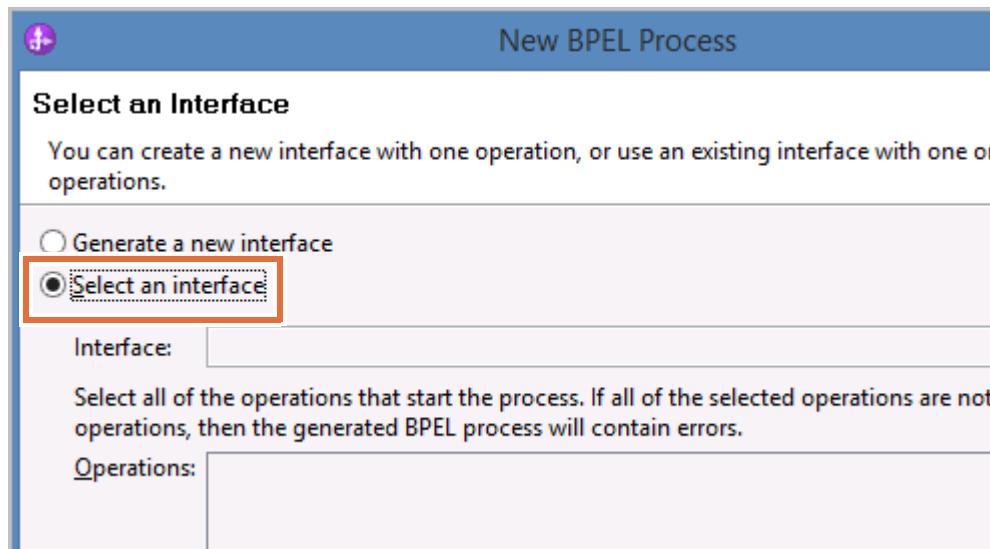


- ___ i. Click **Next**.
- ___ j. At the **BPEL Process Type Selection** screen, leave **Long-running process** selected and **Use IBM Process Server extensions** checked.

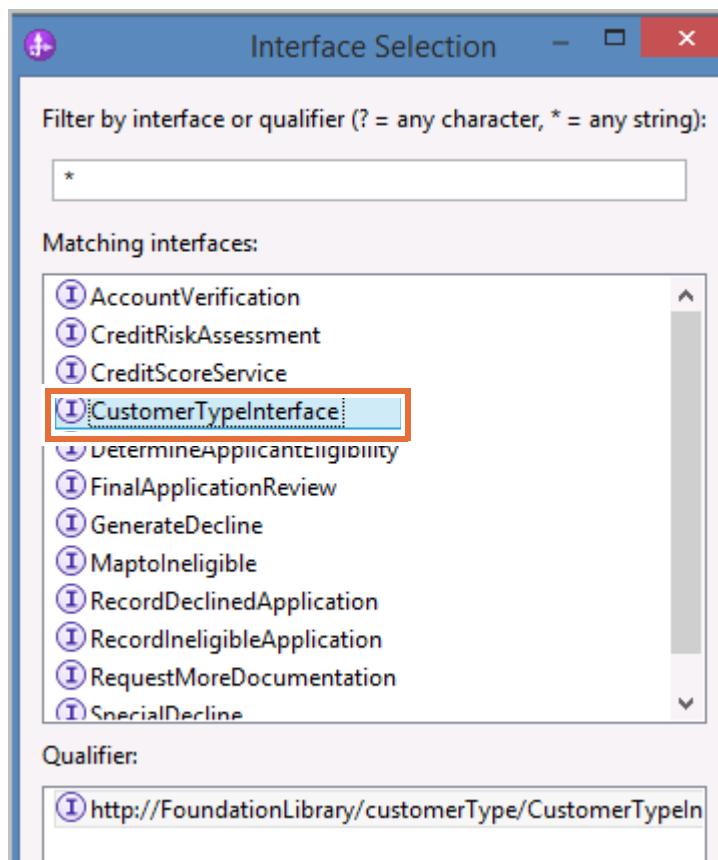


- ___ k. Click **Next**.

- __ l. At the **Select an Interface** screen, click the **Select an Interface** option.

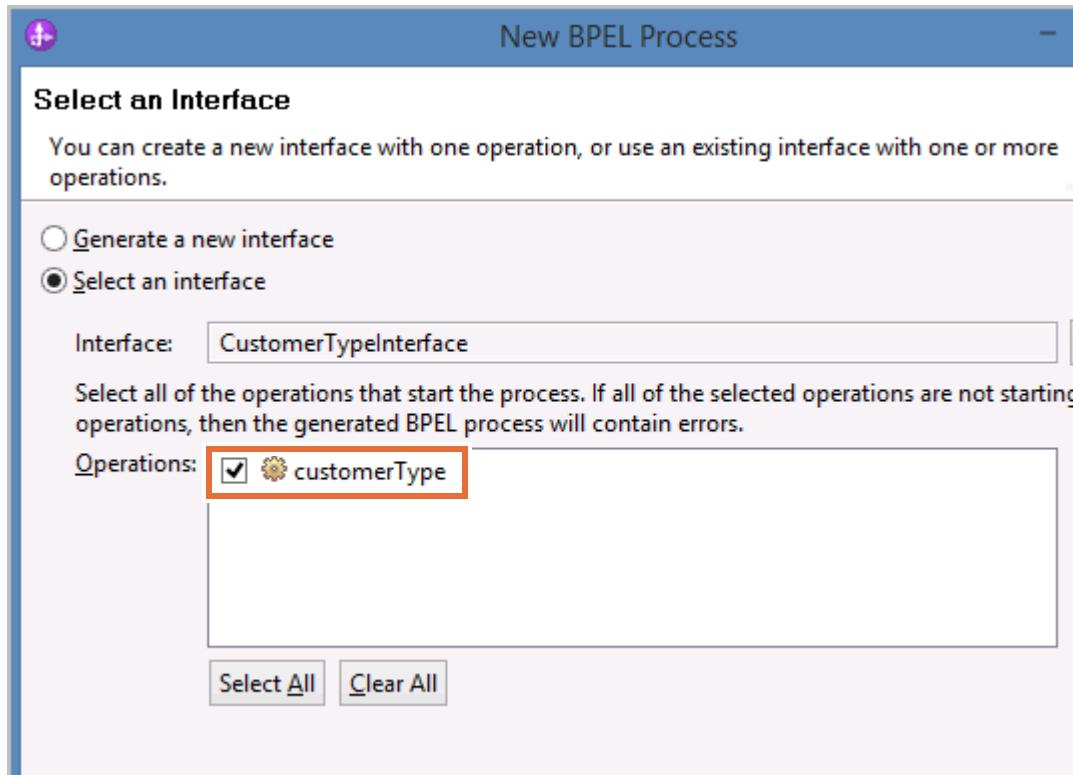


- __ m. For the **Interface** field, click **Browse**.
 __ n. At the **Interface Selection** window, select the **CustomerTypeInterface** from the **Matching interfaces** list.

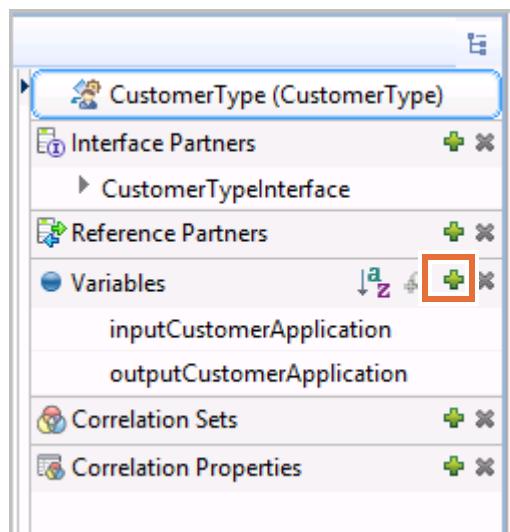


- __ o. Click **OK**.

- ___ p. Verify that the **customerType** operation is checked in the **Operations** window.

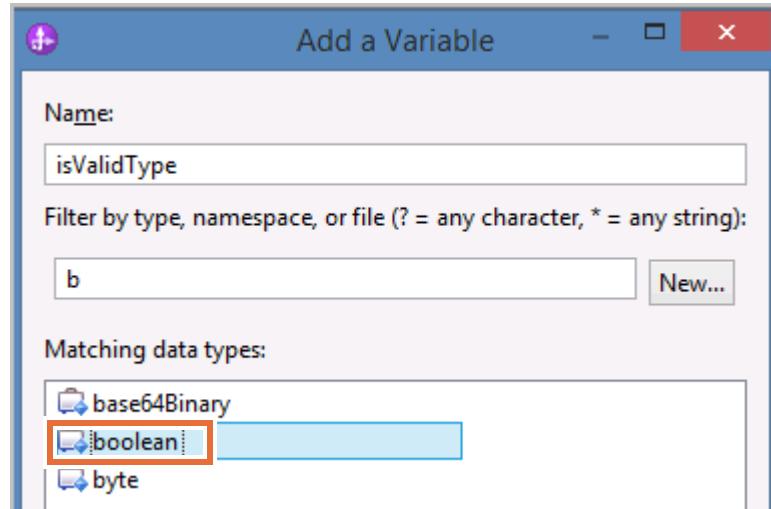


- ___ q. Click **Finish** to create the **CustomerType** business process.
- ___ 2. Define a new **isValidType** variable that is needed in the business process logic.
- ___ a. In the tray, click the **Add a global Variable** (+) icon to create a variable for the BPEL process.

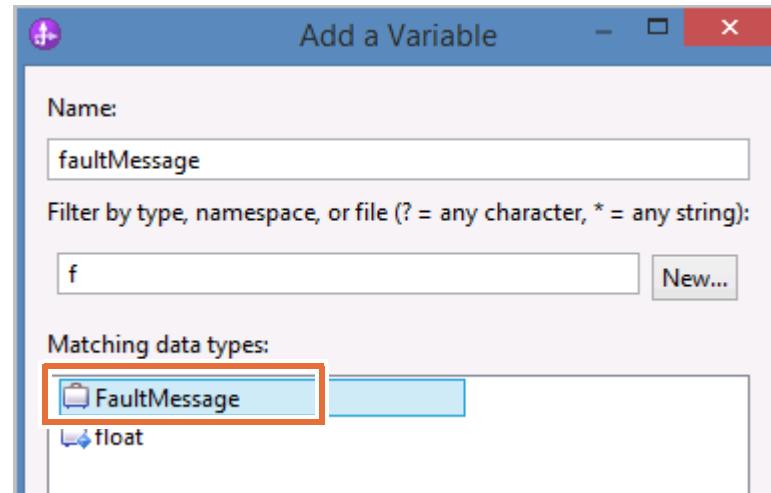


- ___ b. In the **Add a Variable** window, change the default variable **Name** to: **isValidType**
- ___ c. Type **b** in the **Filter by type, namespace, or file** (? = any character, * = any String) field.

- ___ d. Select **boolean** from the **Matching data types** list.

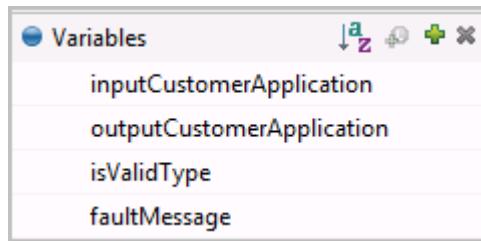


- ___ e. Click **OK**.
- ___ 3. Define a new **FaultMessage** variable that is needed in the business process logic.
- ___ a. Click the **Add a global Variable** (+) icon to create a variable for the BPEL process.
- ___ b. Change the default **Variable** name to: **faultMessage**
- ___ c. In the **Filter by type** prompt, type **f** to filter matching data types.
- ___ d. Select the **FaultMessage** data type.

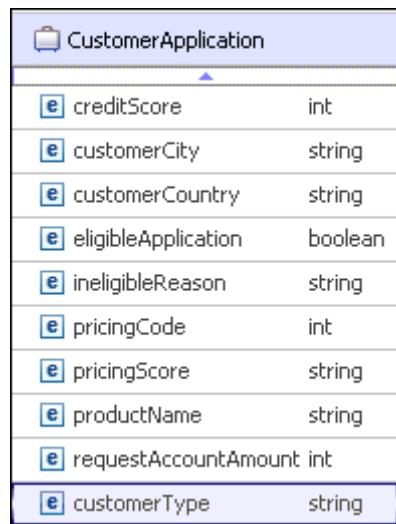


- ___ e. Click **OK**.

- ___ 4. Save your changes. Four variables are now listed in the tray: **outputCustomerApplication**, **faultMessage**, **inputCustomerApplication**, and **isValidType**.

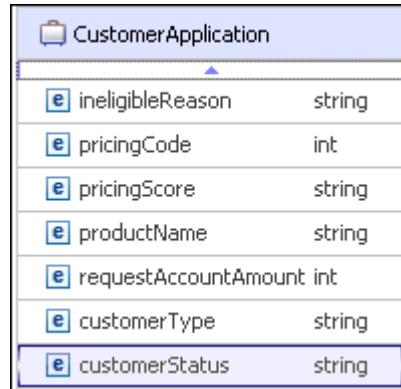


- ___ 5. Add the **customerType** attribute to the **CustomerApplication** business object.
- In the **Business Integration** view, expand **FoundationLibrary > Data > creditserviceitems**.
 - Double-click **CustomerApplication**.
 - In the business object editor, click the **Add a field to a business object** icon.
 - Change the name from `field1` to: `customerType`
 - Leave the data type set to: `string`



- ___ 6. Add the **customerStatus** attribute to the **CustomerApplication** business object.
- In **CustomerApplication** business object editor, click the **Add a field to a business object** icon.
 - Change the name from `field1` to: `customerStatus`

- ___ c. Leave the data type set to: string



- ___ d. Save your changes and close the business object editor.

Part 4: Implementing the CustomerType business process logic

In this part of the exercise, you implement the BPEL logic for the **CustomerType** business process you defined. The purpose of this business process is to distinguish between customer types and to reject anything other than “Customer.” This process would also lend itself to a design where questionable or erroneous data might receive human intervention for validation or correction.

The **CustomerType** BPEL process separates any data validation logic (and possible associated human interaction) from the existing **AccountVerification** BPEL process. After data validation, you implement two types of reply activities, which are based on the validation outcome: reply normal and reply fault. The reply fault activity returns a fault message to the requesting service instead of the standard business data that is associated with reply normal.

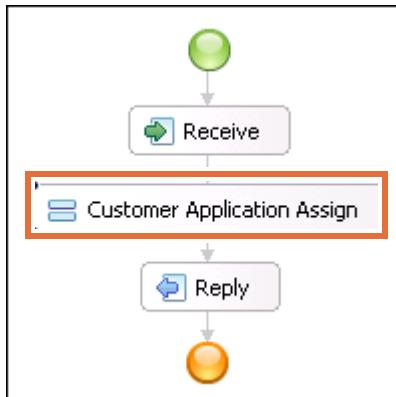


Note

This part of the exercise is concerned with demonstrating the implementation of handlers. Therefore, the **CustomerType** business process is greatly simplified for the sake of brevity. A more appropriate data validation process would use externalized business rules, and automated or human data validation.

- ___ 1. Add a new **Assign** activity to the process to map the **CustomerTask** service output variable to a process variable for an outgoing request call.
 - ___ a. In the **CustomerType** business process editor palette, select the **Assign** icon.
 - ___ b. Click under the **Receive** activity to add the **Assign** to the business process.

- ___ c. Change the display name to: Customer Application Assign

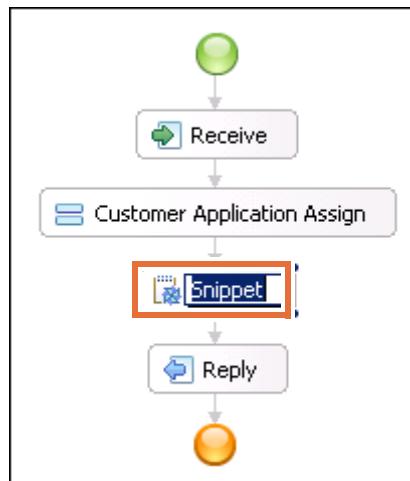


- ___ d. Click the **Customer Application Assign** activity, and select the **Properties > Details** tab.
- ___ e. In the **Assign From** list, click **Select From**.
- ___ f. From the list, select the **inputCustomerApplication** variable.
- ___ g. In the **Assign To** list, click **Select To**.
- ___ h. From the list, select the **outputCustomerApplication** variable.

Assign - Customer Application Assign (Assign)			
Description	Assign From	Assign To	
Details	<input checked="" type="radio"/> inputCustomerApplication	<input checked="" type="radio"/> outputCustomerApplication	
Server			
Exit Condition			

- ___ i. Press **Ctrl+S** to save your changes.
- ___ 2. Define a **Customer Type Validation** snippet.
- ___ a. In the **CustomerType** business process editor palette, in the **Basic Actions** drawer, click the **Snippet** icon.

- b. Click under the **Customer Application Assign** activity to add the **Snippet** to the business process.



- c. Change the display name of the snippet to: Customer Type Validation
 d. Select the **Customer Type Validation** snippet.
 e. In the **Properties** view for the **Customer Type Validation** snippet activity, click the **Details** tab.
 f. Click the **Java** option to indicate that the snippet logic is going to be defined directly as Java code.
 g. If you receive the **Question** prompt, click **Yes**.
 h. Open Windows Explorer and change directory to: C:\labfiles\Support Files\EX4
 i. Open EX4_Java_Snippets.txt in a text editor such as Notepad. This file contains all of the Java code that the different parts of your process use.
 j. Copy CustomerType >>> Customer Type Validation Snippet to the Customer Type Validation snippet. You do not need to copy the comment lines that begin with //.

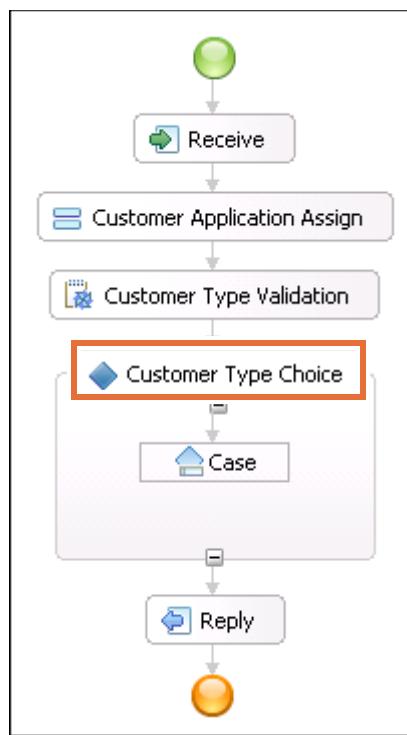
```

// CustomerType >>> Customer Type Validation Snippet
isValidType = new Boolean(false);
String s = new String(inputCustomerApplication.getString("customerType"));
if (s.compareToIgnoreCase("Customer") == 0)
    isValidType = new Boolean(true);
System.out.println(">>> CustomerType >>> Customer Type Snippet: " +
    inputCustomerApplication.getString("accountNumber") + " - " +
    inputCustomerApplication.getString("companyName") + " - " +
    inputCustomerApplication.getString("customerType") + " - " +
isValidType.toString());
  
```

- ___ k. Paste the snippet in the expression window over the existing text. Alternatively, you can manually replace the code with the following code:

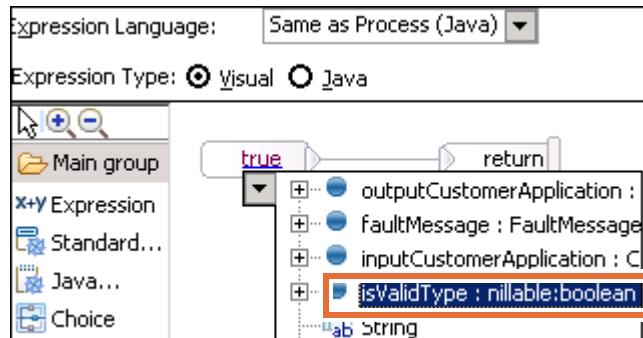
```
isValidType = new Boolean(false);
String s = new
String(inputCustomerApplication.getString("customerType"));
if (s.compareToIgnoreCase("Customer") == 0)
isValidType = new Boolean(true);
System.out.println(">>> CustomerType >>> Customer Type Snippet: " +
inputCustomerApplication.getString("accountNumber") + " - " +
inputCustomerApplication.getString("companyName") + " - " +
inputCustomerApplication.getString("customerType") + " - " +
isValidType.toString());
```

- ___ l. Save your changes.
- ___ 3. Define the **Customer Type Choice**. Based on the validation of the customer type that the **Customer Type Validation** snippet determines, the choice initiates the appropriate reply activity; either call a snippet or throw a fault.
- ___ a. In the business process editor palette, in the **Structures** drawer, click the **Choice** icon.
 - ___ b. Click below the **Customer Type Validation** snippet to paste in a new **Choice** construct.
 - ___ c. Change the **Display name** to: Customer Type Choice

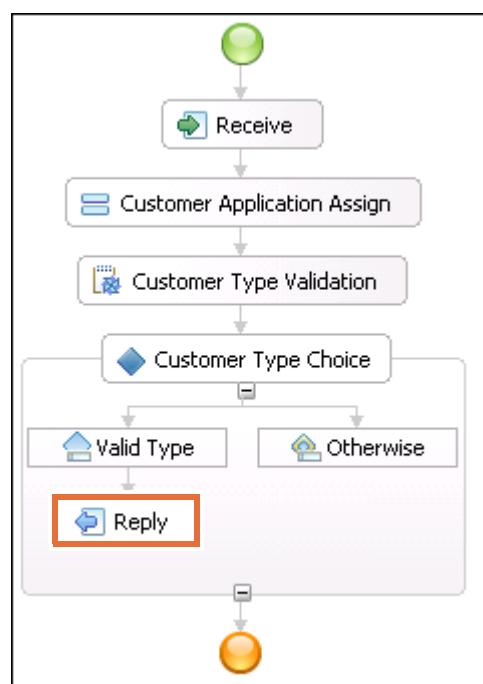


- ___ d. Click the **Case** construct within the **Customer Type Choice** component.
- ___ e. From the **Properties > Description** tab, change the **Display name** to: Valid Type
- ___ f. Click the **Properties > Details** tab for the valid type case construct.
- ___ g. In the **Expression Language** field, select **Same as Process (Java)** from the list.

- ___ h. In the visual snippet editor, click inside the **true** expression snippet.
- ___ i. Select the **isValidType** variable from the list and press the Enter key. The expression returns the isValidType Boolean variable.

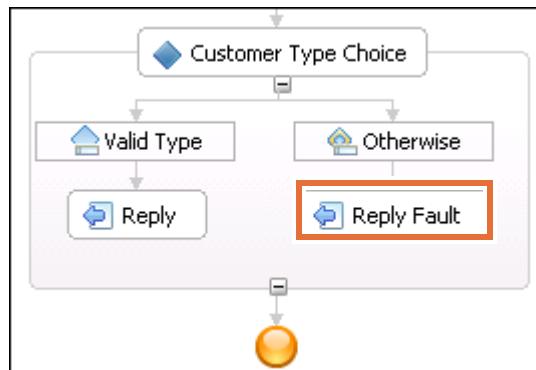


- ___ j. In the business process editor, right-click **Customer Type Choice** and choose **Add an Otherwise Element** from the menu. It adds an **Otherwise** construct.
- ___ 4. Move the existing **Reply** activity and add a **Reply Fault** activity to the process.
- ___ a. In the editor, select the existing **Reply** activity, and drag it under the **Valid Type** case.



- ___ b. With the **Reply** node in focus, click **Properties > Details**. Notice that the reply details, such as Partner, Interface, and Operation, are already entered; and the **Reply Type** is **Normal**.
- ___ c. In the business process editor palette, in the **Basic Actions** drawer, click the **Reply** icon.
- ___ d. Click under the **Otherwise** case construct to add a **Reply** activity.

- __ e. Change the display name to: Reply Fault

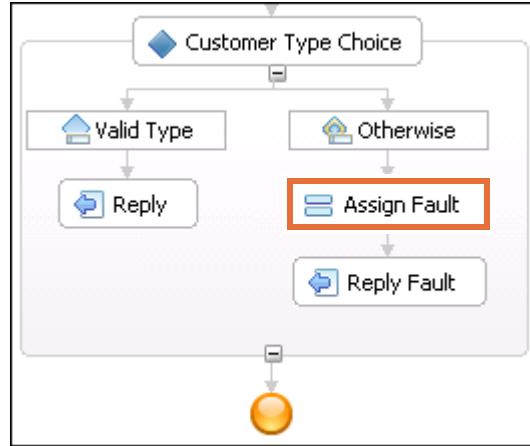


- __ f. Click the **Reply Fault** node to give it focus. In the **Details** tab of the **Properties** view, click **Browse** to select a new partner.
- __ g. From the **Select a Partner** window, select **CustomerTypeInterface**.
- __ h. Click **OK**.
- __ i. The **Interface** field is automatically entered as **CustomerTypeInterface**.
- __ j. The **Operation** field is automatically entered as **customerType**.
- __ k. Ensure that **Use data type variables mapping** is selected.
- __ l. Click the **Fault** option to enable selection of the appropriate **Reply Type** and leave **Fault Name** as **faultMessage**.
- __ m. For the **Fault(s)** variable, in the **Variable** column, click **(none)**.
- __ n. Select the **faultMessage** variable.

Partner: *	CustomerTypeInterface	Reply Type:	<input type="radio"/> Normal <input checked="" type="radio"/> Fault
Interface: *	CustomerTypeInterface	Fault Name:	faultMessage
Operation: *	customerType	<input checked="" type="checkbox"/> Use data type variables mapping	
Fault(s)	Name	Type	Read from Variable
	customerTypeFault1_faultMessage	FaultMessage	faultMessage

- __ o. Save your changes. The Problems view must have no errors in it.
- __ 5. To return a meaningful fault reply, the fault message must be populated with data. Add a new **Assign** activity to the process to initialize the fault variable.
- __ a. In the business process editor palette, in the **Basic Actions** drawer, click the **Assign** icon.
- __ b. Click under the **Otherwise** case construct to paste a new **Assign** activity right above the already present **Reply Fault** activity.

- ___ c. Change the display name to: Assign Fault



- ___ d. Click the **Assign Fault** activity and click the **Properties > Details** tab.
- ___ e. In the **Assign From** column, click **Select From**.
- ___ f. From the list of variables, expand the **outputCustomerApplication**, and select the **companyName:string** attribute.
- ___ g. In the **Assign To** list, click **Select To**.
- ___ h. From the variable list, expand the **faultMessage:FaultMessage** variable and click the **FaultID:string** attribute to select it.
- ___ i. Click **Add** to add a second assignment.
- ___ j. In the **Assign From** column, click **Select From**.
- ___ k. From the list, expand **outputCustomerApplication** and select the **customerType:string** variable.
- ___ l. In the **Assign To** column, click **Select To**.
- ___ m. From the list, expand the **faultMessage:FaultMessage** variable.
- ___ n. From this variable, select the **FaultData:string** attribute.

Assign From	Assign To
outputCustomerApplication companyName	FaultMessage FaultID
outputCustomerApplication customerType	FaultMessage FaultData

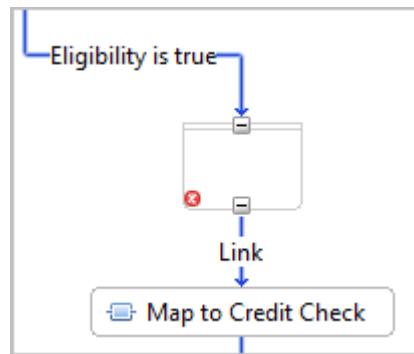
- ___ o. Save your changes and close the business process editor.

Part 5: Integrating CustomerType with the AccountVerification process

In this part of the exercise, you modify the **AccountVerification** BPEL process to use the new **CustomerType** business process to validate the customer type. To support the addition of fault handling, the **CustomerType** business process checks a customer type and returns a Reply or Fault. The returned Reply or Fault depends on the type of customer that is contained in the input

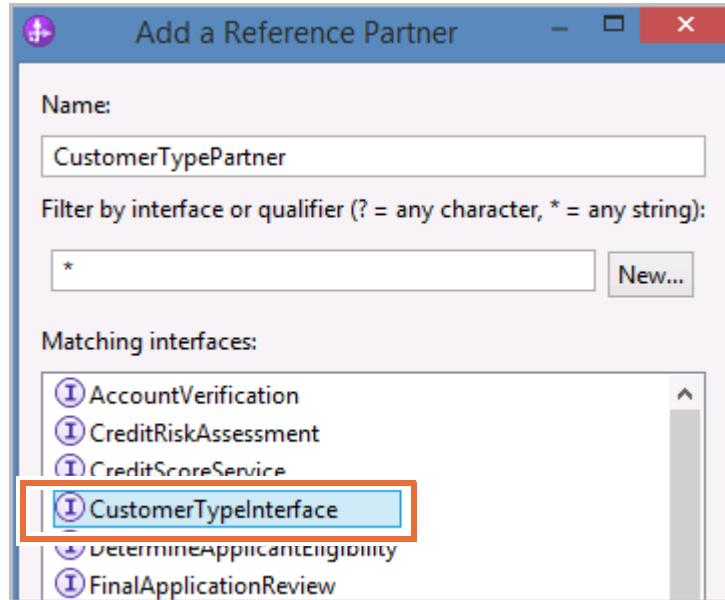
business object. Upon calling this partner and receiving a result, fault handling is included in **AccountVerification** to support a possible Fault return from **CustomerType**.

- ___ 1. Create the scope that contains the customer type validation in the **AccountVerification** process.
 - ___ a. In the **Business Integration** view, expand **FoundationModule > Integration Logic > BPEL Processes > AccountVerification**.
 - ___ b. Double-click the **AccountVerification** process to open it in the business process editor.
 - ___ c. Right-click in the white space inside **AccountVerification_Flow** and select **Arrange Parallel Activities Contents**.
 - ___ d. In the **Palette**, in the **Structures** drawer, click **Scope**, and drop it on the canvas inside **AccountVerification_Flow** above the **Map to Credit Check** activity.
 - ___ e. Click **Scope**. On the **Properties > Description** tab, change **Display name** to: **Type Validation Scope**
 - ___ f. Click the link that leads between **Determine Application Eligibility** and **Map to Credit Check**.
 - ___ g. Hover the mouse pointer over the connector where the link meets the **Map to Credit Check** activity. The pointer turns into a crosshair (two intersecting, double-headed arrows).
 - ___ h. Drag the connector onto the scope activity.
 - ___ i. Right-click **Type Validation Scope** and choose **Add a link** from the menu.
 - ___ j. Click the **Map to Credit Check** activity to add the link.
 - ___ k. Optional: Right-click in the white space inside **AccountVerification_Flow** and select **Arrange Parallel Activities Contents**.
 - ___ l. Save your changes.



- ___ 2. Define a new **CustomerTypePartner** to reference the **CustomerType** BPEL process.
 - ___ a. In the **Reference Partners** section of the tray, click **Add a Reference Partner**.
 - ___ b. In the **Add Reference Partner** window, change the **Name** to: **CustomerTypePartner**

- __ c. From the **Matching Interfaces**, select **CustomerTypeInterface**.



- __ d. Click **OK**.
- __ 3. Call the partner type validation service.
- __ a. In the business process editor palette, in the **Basic Actions** drawer, click the **Invoke** icon.
 - __ b. Click in the scope construct to add the invoke activity.
 - __ c. Change the display name to: **Customer Type Invoke**
- A diagram showing a rectangular scope construct. Inside the scope, there is a single activity node with a green diamond icon and the text "Customer Type Invoke".
- __ d. Select the **Customer Type Invoke** activity.
 - __ e. Select the **Properties > Details** tab.
 - __ f. Click **Browse** to select a **Partner**.
 - __ g. In the **Select a Partner** window, select **CustomerTypePartner**.
 - __ h. Click **OK**.
 - __ i. The **Interface** field is automatically set to **CustomerTypeInterface**.
 - __ j. The **Operation** field is automatically set to **customerType**.
 - __ k. Ensure that the **Use data type variables mapping** check box is selected.
 - __ l. For the **Input(s)** variable, click the **(none)** link.
 - __ m. Select the **CustomerApplicationVariable** variable.
 - __ n. For the **Output(s)** variable, click the **(none)** link.

- __ o. Select the **CustomerApplicationVariable** variable.

Partner*: CustomerTypePartner [Browse...](#)

Interface*: CustomerTypeInterface

Operation*: customerType

Use data type variables mapping

	Name	Type
Inputs	inputCustomerApplication	CustomerApplication
	Name	Type
Outputs	outputCustomerApplication	CustomerApplication

Read from Variable
CustomerApplicationVariable

Store into Variable
 CustomerApplicationVariable

- __ p. Select the **Server** tab.

- __ q. Change the **Continue processing upon unhandled faults** option to **Yes**.

Transactional Behavior

Specify the transactional behavior for the selected activity. This behavior might differ if, for example, the activity is part of a larger transaction.

Commit before The preceding transaction commits before this activity starts.

Commit after The transaction commits immediately after this activity has completed.

Participates The activity runs in an existing transaction.

Requires own The activity is isolated in its own transaction.

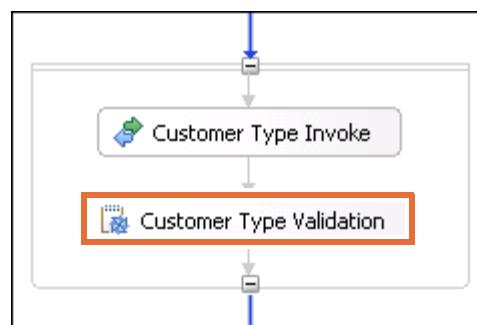
Enable persistence and queries of business-relevant data

Continue processing upon unhandled faults: Same as Process Yes No

- __ r. Save your changes.

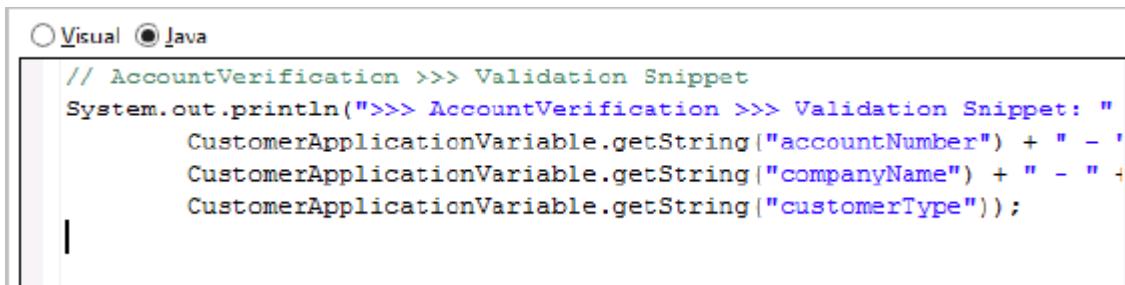
4. Create a **Customer Type Validation** snippet.

- __ a. In the **Palette**, in the **Basic Actions** drawer, click **Snippet**.
- __ b. Place the snippet in the **Type Validation Scope**, below the **Customer Type Invoke**.
- __ c. Change the snippet display name to: **Customer Type Validation**



- __ d. In the **Properties** view for the **Customer Type Validation** snippet activity, click the **Details** tab.

- ___ e. Click the **Java** option to indicate that the snippet logic is going to be defined directly as Java code.
- ___ f. If you receive a **Question** prompt, click **Yes**.
- ___ g. Copy AccountVerification >>> Validation Snippet to the Customer Type Validation snippet from C:\labfiles\Support Files\Ex4\EX4_Java_Snippets.txt. You do not need to copy the comment lines that begin with //.



The screenshot shows a code editor window with two tabs at the top: 'Visual' (unchecked) and 'Java' (checked). The code in the editor is:

```
// AccountVerification >>> Validation Snippet
System.out.println(">>> AccountVerification >>> Validation Snippet: "
    + CustomerApplicationVariable.getString("accountNumber") + " - "
    + CustomerApplicationVariable.getString("companyName") + " - " +
    CustomerApplicationVariable.getString("customerType"));
```

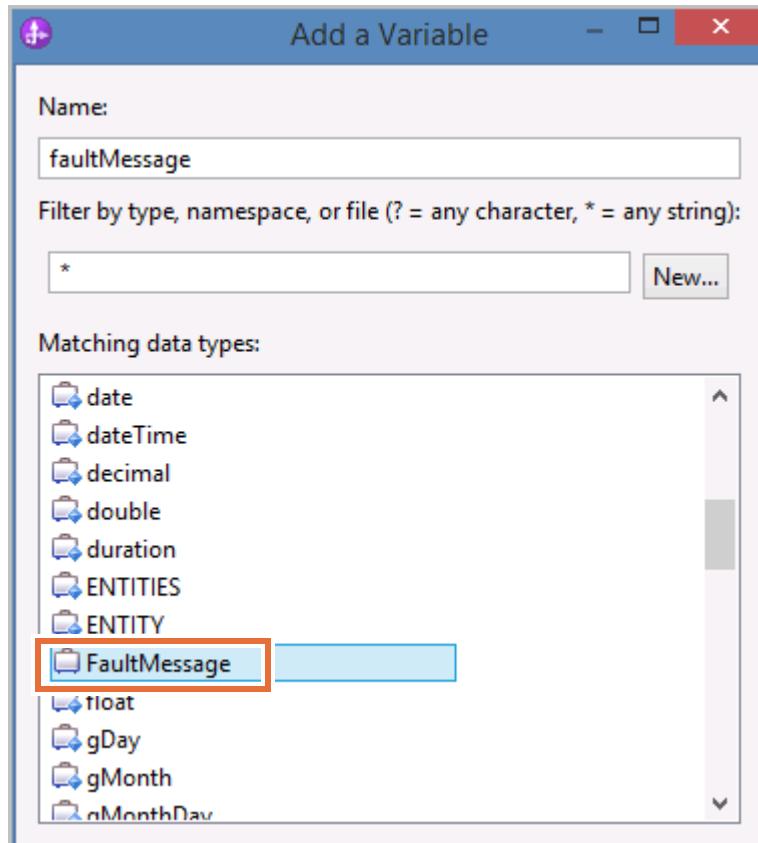
- ___ h. Paste the snippet in the expression window over the existing text. Alternatively, you can manually replace the code with the following code:
- ```
System.out.println(">>> AccountVerification >>> Validation Snippet: " +
 CustomerApplicationVariable.getString("accountNumber") + " - " +
 CustomerApplicationVariable.getString("companyName") + " - " +
 CustomerApplicationVariable.getString("customerType"));
```
- \_\_\_ i. Save your changes.
  - \_\_\_ j. Right-click the process editor and choose **Align Parallel Activities Contents Automatically**.

## **Part 6: Adding a Fault Handler to AccountVerification**

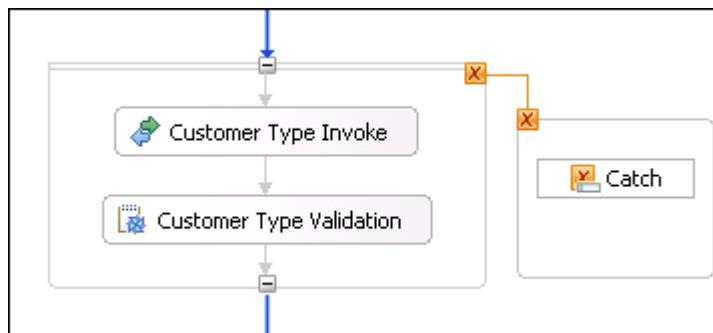
You now add appropriate fault handling in the **AccountVerification** business process to support the addition of fault handling for the **CustomerType** business process. This business process checks a customer type and returns a **Reply** or **Fault** depending on the type of customer that is contained in its input business object. Upon calling on this partner and receiving a result, fault handling must be made available in **AccountVerification**.

- \_\_\_ 1. Define a new **faultMessage** variable in the **AccountVerification** process so that the process can receive a fault reply, possibly returned from another service.
  - \_\_\_ a. In the business process editor, click the **AccountVerification(AccountVerification)** process icon in the upper-right corner of the tray. (The purpose of this step is to move the focus of the tool away from the **Type Validation Scope**.)
  - \_\_\_ b. Under **Variables**, click the **Add a global Variable (+)** icon to add process variable.
  - \_\_\_ c. In the **Add a Variable** window, change the default **Name** to: **faultMessage**

- \_\_\_ d. In the **Matching data types** window, select **FaultMessage**.



- \_\_\_ e. Click **OK**.
- \_\_\_ 2. Implement a new fault handler on the **Type Validation Scope**. It catches an explicit user-type fault **FaultMessage** and rethrows it to the next enclosing scope.
- In the business process editor, right-click the **Type Validation Scope** and select **Add a Fault Handler** from the menu (or click the **Add a Fault Handler** icon from the scope menu).
  - A new fault handler is shown beside the **Type Validation Scope**.



- Click the **Catch** construct within the new fault handler.
- Change the default **Name** to: `faultMessage`

- \_\_ e. From the **Properties > Details** tab, ensure that for **Fault Type** the **User-defined** option is selected, since you want this particular fault handler to catch an explicit user-type fault.
- \_\_ f. For **Fault Name**, select **{http://FoundationLibrary/customertype/CustomerTypeInterface}faultMessage** from the list.
- \_\_ g. Enter **faultMessage** in the **Variable Name** field.

|                |                                                                              |
|----------------|------------------------------------------------------------------------------|
| Fault Type:    | <input type="radio"/> Built-in <input checked="" type="radio"/> User-defined |
| Fault Name:    | {http://FoundationLibrary/customertype/CustomerTypeInterface}faultMessage    |
| Variable Name: | faultMessage                                                                 |

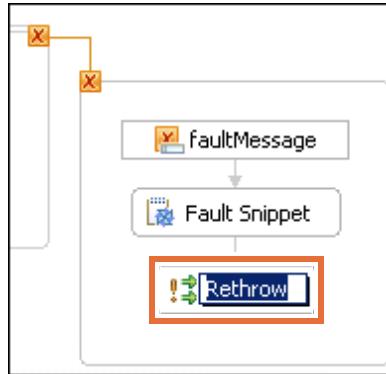
- \_\_ h. Save your changes.
- \_\_ 3. Add a snippet to the **Fault Handler** to print the fault message to the server log.
  - \_\_ a. In the business process editor palette, in the **Basic Actions** drawer, select the **Snippet** icon.
  - \_\_ b. Click under the **faultMessage** catch construct to add a snippet activity.
  - \_\_ c. Change the display name to: **Fault Snippet**
  - \_\_ d. In the **Properties** view for the fault snippet activity, select the **Details** tab.
  - \_\_ e. Select the **Java** option to indicate that the snippet logic is defined in Java code.
  - \_\_ f. If you receive a Question prompt, click **Yes**.
  - \_\_ g. Copy **AccountVerification >>> Type Validation Scope >>> Catch faultMessage Snippet** to the Customer Type Validation snippet from **C:\labfiles\Support Files\EX4\EX4\_Java\_Snippets.txt**. You do not need to copy the comment lines that begin with **//**.

```
// AccountVerification >>> Type Validation Scope >>> Catch faultMessage Snippet
System.out.println(">>> AccountVerification >>> Type Validation Scope >>> Catch
faultMessage: " +
 faultMessage.getString("FaultID") + " - " + faultMessage.getString
("FaultData"));
```

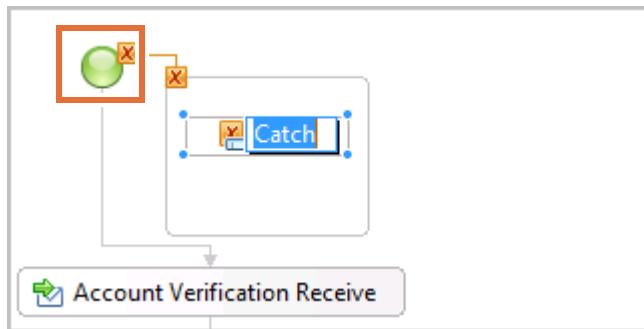
- \_\_ h. Paste the snippet in the expression window over the existing text. Alternatively, you can manually replace the code with the following code:
 

```
System.out.println(">>> AccountVerification >>> Type Validation Scope >>>
Catch faultMessage: " +
 faultMessage.getString("FaultID") + " - " +
 faultMessage.getString("FaultData"));
```
- \_\_ i. Save your changes.
- \_\_ 4. Add a rethrow activity to the **faultMessage** catch construct.
  - \_\_ a. In the business process editor palette, in the **Faults** drawer, click **Rethrow**.

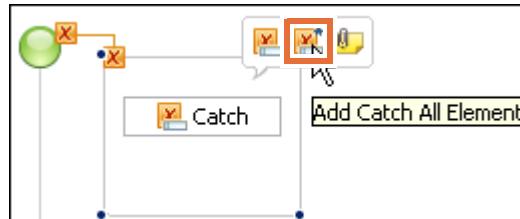
- \_\_\_ b. Click under **Fault Snippet** to add the **Rethrow** activity to the **Fault Handler**.



- \_\_\_ c. Accept the default display name.
- \_\_\_ 5. Implement a new default fault handler on the top-level process scope. It catches all types of faults that are propagated to it, and then it terminates the process instance.
- \_\_\_ a. In the business process editor, right-click the start node (a green circle) and choose **Add a Fault Handler** from the menu. A new **Fault Handler** is shown beside the start node, which is also the top-level process scope.



- \_\_\_ b. Click the new **Fault Handler**.
- \_\_\_ c. Select **Add Catch All Element** from the pop-up icon bar.

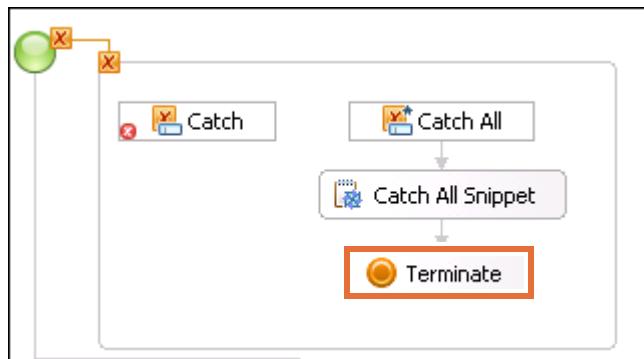


- \_\_\_ d. In the business process editor palette, in the **Basic Actions** drawer, select the **Snippet** icon.
- \_\_\_ e. Drop the new **Snippet** activity under **Catch All** in the **Fault Handler**.
- \_\_\_ f. Change the display name to: **Catch All Snippet**
- \_\_\_ g. In the **Properties** view, select the **Details** tab.
- \_\_\_ h. Select the **Java** option to indicate that the snippet logic is defined as Java code.
- \_\_\_ i. If you receive a Question prompt, click **Yes**.

- \_\_\_ j. Enter the following Java code in the window or copy AccountVerification >>> Default Fault Handler >>> Catch All Snippet to the Catch All Snippet snippet from C:\labfiles\Support Files\EX4\EX4\_Java\_Snippets.txt. You do not need to copy the comment lines that begin with //.

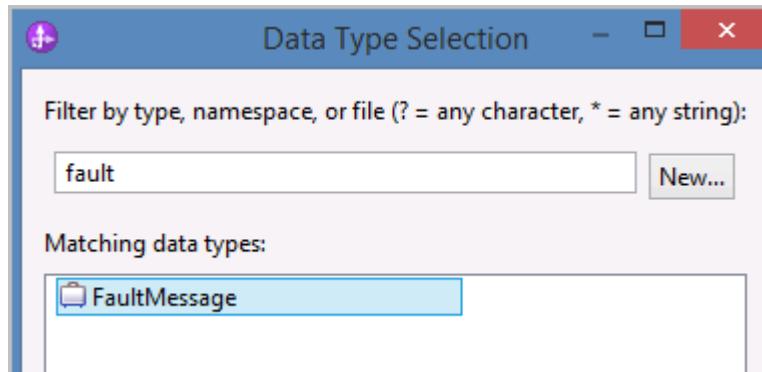
```
System.out.println(">>> AccountVerification >>> Default Fault Handler >>>
Catch All");
```

- \_\_\_ k. In the business process editor palette, in the **Faults** drawer, select **Terminate**.  
 \_\_\_ l. Add the **Terminate** activity under the **Catch All Snippet** in the **Fault Handler**.  
 \_\_\_ m. Accept the default display name.  
 \_\_\_ n. Save your changes.

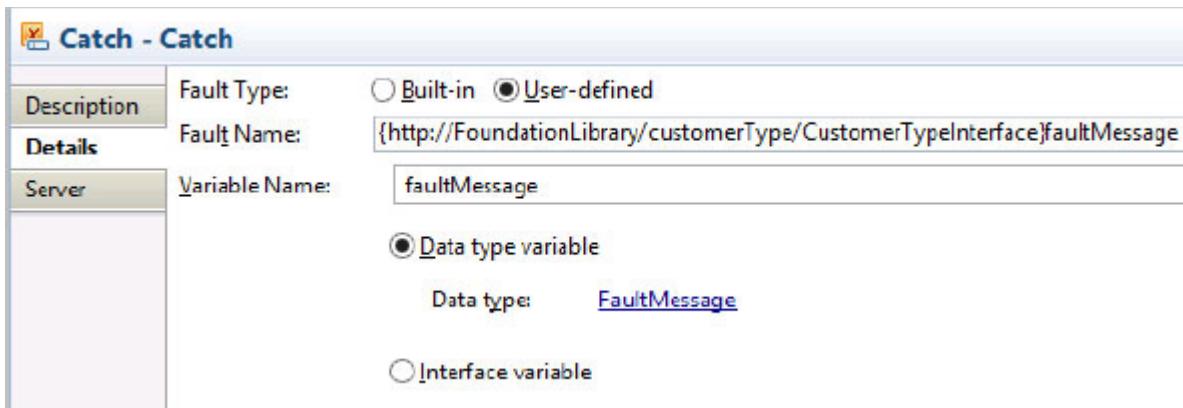


- \_\_\_ 6. Implement the **Catch** block for the **faultMessage** fault type in the top-level fault handler.
- \_\_\_ a. Click the **Catch** catch construct within the top-level fault handler.
  - \_\_\_ b. Click the **Properties > Details** tab.
  - \_\_\_ c. Set the **Fault Type** option to **User-defined** since this particular fault handler catches an explicit user-type fault.
  - \_\_\_ d. From the **Fault Name** list, select {<http://FoundationLibrary/customertype>}**CustomerTypeInterface****faultMessage**.
  - \_\_\_ e. Type **faultMessage** in the **Variable Name** field.
  - \_\_\_ f. Set the variable type to **Data type variable**.
  - \_\_\_ g. Click **Browse** to select an appropriate data type.
  - \_\_\_ h. At the **Data Type Selection** prompt, enter **fault** in the **Filter by type** field.

- \_\_ i. Select the matching **FaultMessage** data type entry.

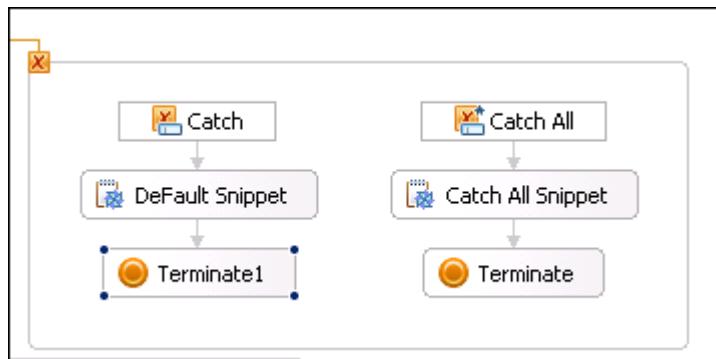


- \_\_ j. Click **OK**.  
\_\_ k. Save your changes.



- \_\_ 7. Add a snippet to the **Fault Handler** to print the fault message to the server log.
- \_\_ a. In the business process editor palette, in the **Basic Actions** drawer, click **Snippet**.  
\_\_ b. Click under the **Catch** catch construct to add a snippet activity.  
\_\_ c. Change the display name to: **DeFault Snippet**  
(Note the case.)  
\_\_ d. In the **Properties** view for the DeDefault snippet activity, select the **Details** tab.  
\_\_ e. Select the **Java** option to indicate that the snippet logic is defined in Java code.  
\_\_ f. If you receive a Question prompt, click **Yes**.  
\_\_ g. Enter the following Java code in the window or copy AccountVerification >>> Default Fault Handler >>> Catch faultMessage Snippet to the DeDefault Snippet snippet from C:\labfiles\Support Files\EX4\EX4\_Java\_Snippets.txt. You do not need to copy the comment lines that begin with //.
- ```
System.out.println(">>> AccountVerification >>> Default Fault Handler >>>
Catch faultMessage: " +
faultMessage.getString("FaultID") + " - " +
faultMessage.getString("FaultData"));
```
- __ h. Save your changes.

- ___ 8. Add a new **Terminate** activity at the end of the **faultMessage** catch construct.
 - ___ a. In the business process editor palette, in the **Faults** drawer, select the **Terminate** icon.
 - ___ b. Click under the **DeFault Snippet** activity to add a new **Terminate** activity.



- ___ c. Accept the default display name.
- ___ 9. Save your changes and close the process editor.

Part 7: Returning a **FaultMessage**

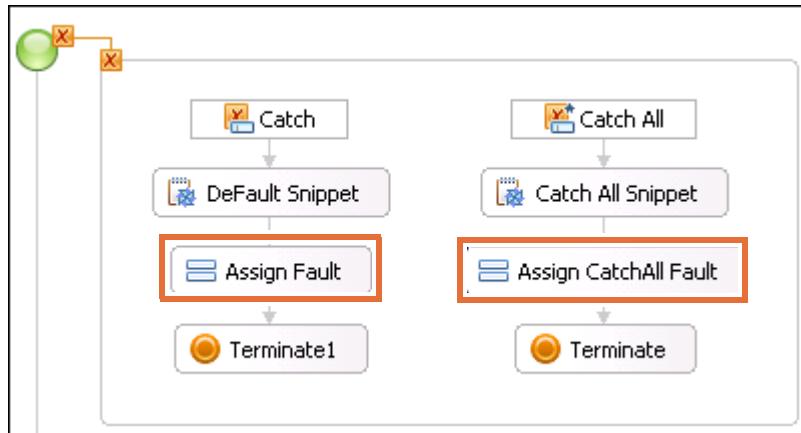
As implemented, the fault handlers terminate the process instance at the top level. It ultimately creates an unhandled exception unless a **FaultMessage** is properly returned after fault handling terminates the process.

- ___ 1. Add a fault to the **AccountVerification** interface.
 - ___ a. In the **Business Integration** view, expand **Foundation Library > Interfaces > accountverification**.
 - ___ b. Double-click the **AccountVerification** interface to open it in the interface editor.
 - ___ c. In the interface editor, click the **InputCriterion** operation.
 - ___ d. On the toolbar, click the **Add Fault** icon.
 - ___ e. In the **Name** column, change the Name from **InputCriterionFault1** to: **faultMessage**
 - ___ f. Click **string** in the **Type** column and select the **FaultMessage** type.

Operations		
Operations and their parameters		
	Name	Type
InputCriterion		
Inputs	Input	CustomerApplication
Outputs	Output	Message
Fault	faultMessage	FaultMessage

- ___ g. Save your changes.

- __ 2. Assign **FaultMessage** data to the exception handler returns.
- __ a. Click the **AccountVerification** business process editor tab where you worked in the previous part.
- __ b. Select **Assign** from the **Palette**.
- __ c. In the process-level fault handler, add **Assign** before **Terminate1** in the **Catch** block.
- __ d. Rename it to: **Assign Fault**
- __ e. Repeat this procedure for the **CatchAll** block.
- __ f. Name it: **Assign CatchAll Fault**

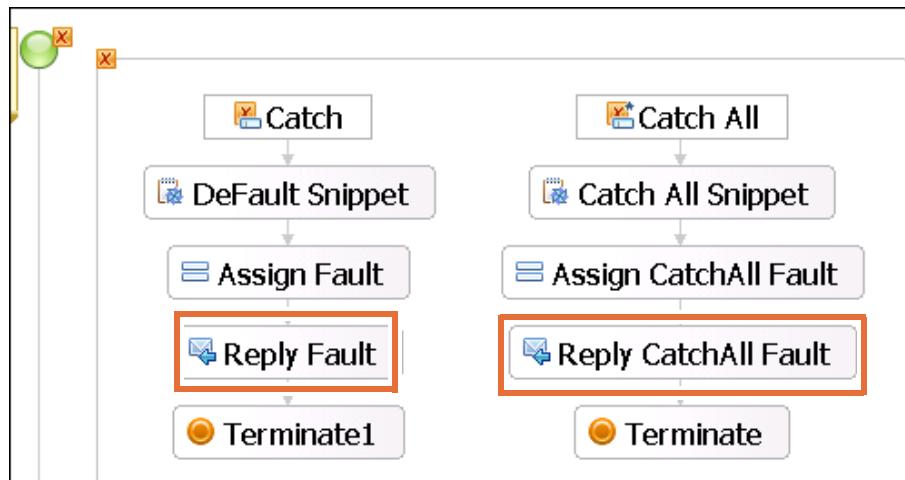


- __ g. Click **Assign Fault**.
- __ h. Click the **Properties > Details** tab.
- __ i. Click **Select From**.
- __ j. Select **CustomerApplicationVariable > companyName**.
- __ k. Click **Select To**.
- __ l. Select **faultMessage > FaultID**.
- __ m. Click **Add** for another assignment.
- __ n. Click **Select From**.
- __ o. Select **CustomerApplicationVariable > customerType**.
- __ p. Click **Select To**.
- __ q. Select **faultMessage > FaultData**.

Assign From	→	Assign To
CustomerApplicationVariable companyName	xy →	FaultMessage FaultID xy
CustomerApplicationVariable customerType	xy →	FaultMessage FaultData xy

- __ r. Click **Assign CatchAll Fault**.
- __ s. Click the **Properties > Details** tab.

- ___ t. Click **Select From**.
 - ___ u. Select **CustomerApplicationVariable > companyName**.
 - ___ v. Click **Select To**.
 - ___ w. Select **faultMessage > FaultID**.
 - ___ x. Click **Add** for another assignment.
 - ___ y. Click **Select From**.
 - ___ z. Select **CustomerApplicationVariable > customerType**.
 - ___ aa. Click **Select To**.
 - ___ ab. Select **faultMessage > FaultData**.
 - ___ ac. Save your changes.
- ___ 3. Reply with **FaultMessage** data from the exception handlers.
- ___ a. From the **Palette**, select **Reply**.
 - ___ b. In the top-level fault handler, add **Reply** before **Terminate1** of the **Catch** block.
 - ___ c. Rename it to: **Reply Fault**
 - ___ d. Repeat this procedure for **CatchAll**.
 - ___ e. Name it: **Reply CatchAll Fault**



- ___ f. Click **Reply Fault**.
- ___ g. Click the **Properties > Details** tab.
- ___ h. For **Partner**, click **Browse**.
- ___ i. From **Select a Partner**, choose **AccountVerification**.
- ___ j. Click **OK**.
- ___ k. For **Reply Type**, click **Fault**.
- ___ l. In the **Read From Variable** column, click **(none)**.

- __ m. Select **faultMessage**.

Partner:*	AccountVerification	Browse...	Reply Type:	<input type="radio"/> Normal <input checked="" type="radio"/> Fault									
Interface:*	AccountVerification		Fault Name:	<input type="text" value="faultMessage"/>									
Operation:*	InputCriterion	▼											
<input checked="" type="checkbox"/> Use data type variables mapping													
<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Read from Variable</th> </tr> </thead> <tbody> <tr> <td> Fault(s)</td> <td>InputCriterionFault1_faultMessage</td> <td>FaultMessage</td> </tr> <tr> <td></td> <td></td> <td>faultMessage</td> </tr> </tbody> </table>					Name	Type	Read from Variable	Fault(s)	InputCriterionFault1_faultMessage	FaultMessage			faultMessage
Name	Type	Read from Variable											
Fault(s)	InputCriterionFault1_faultMessage	FaultMessage											
		faultMessage											

- __ n. Click **Reply CatchAll Fault**.
 __ o. Click the **Properties > Details** tab.
 __ p. For **Partner**, click **Browse**.
 __ q. From **Select a Partner**, choose **AccountVerification**.
 __ r. Click **OK**.
 __ s. For **Reply Type**, click **Fault**.
 __ t. In the **Read From Variable** column, click **(none)**.
 __ u. Select **faultMessage**.
 __ v. Save your changes.

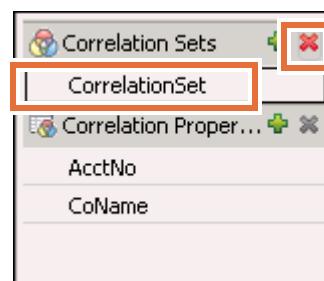


Information

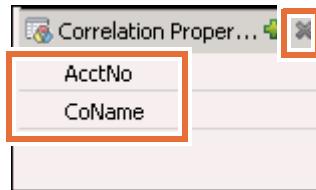
For better legibility and readability, you can hide the fault handler at the process level by clicking the fault icon in the upper-right corner of the start node. To show the handler, click the fault icon again.

Part 8: Define the correlation set and property for AccountVerification

- __ 1. Define a new correlation set for the **AccountVerification** business process, and add new aliases for the **companyName** property for the request, response, and fault messages.
- __ a. In the **Correlation Sets** tray, delete the existing correlation set by selecting it and then clicking the **delete (X)** icon. In the **Delete a Correlation Set** window, click **OK**.

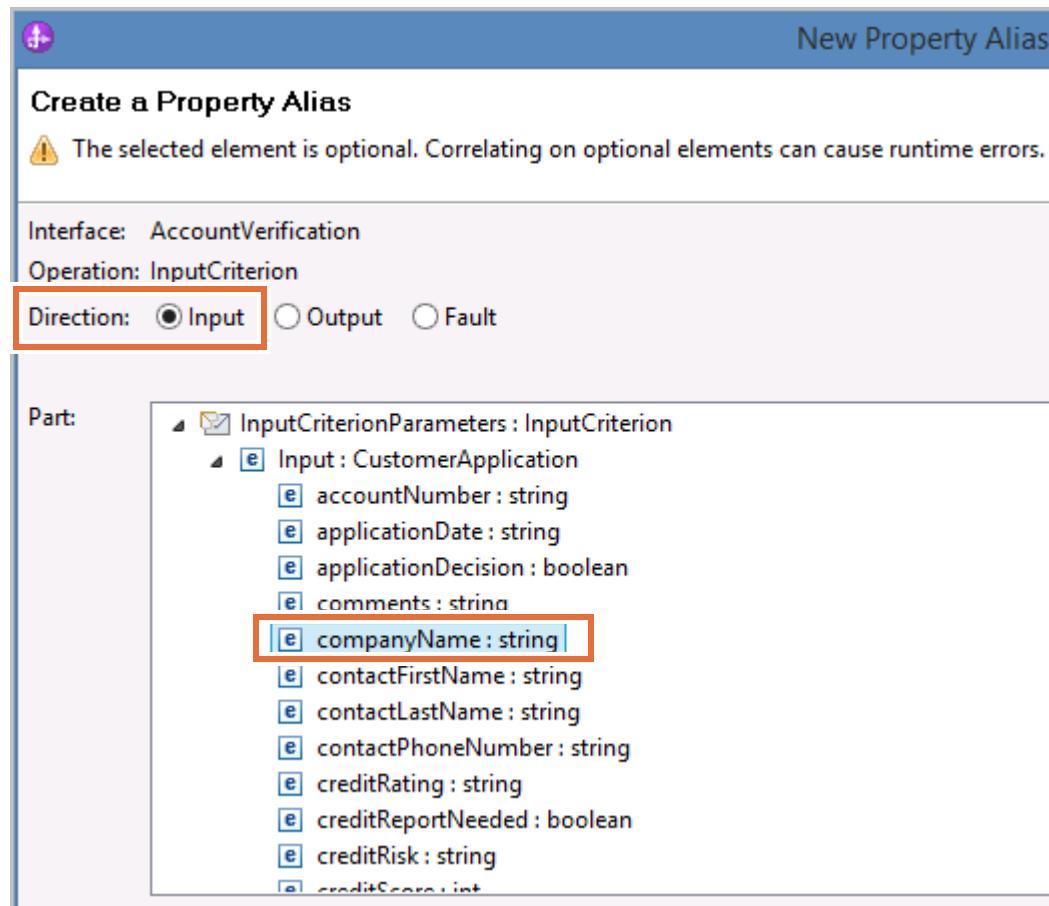


- ___ b. Similarly, delete the correlation properties. Select **AcctNo** and **CoName** properties one at a time and click the **delete (X)** icon. If prompted, in the **Delete a Correlation Property** window, click **OK** each time.



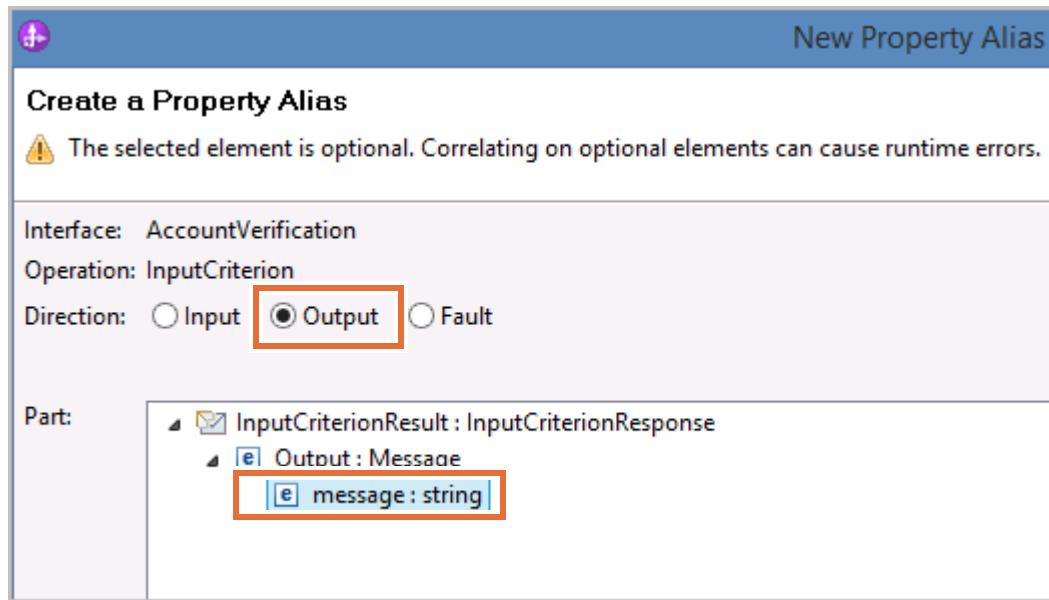
- ___ c. In the business process editor for **AccountVerification**, click the **Add a Correlation Set (+)** icon.
- ___ d. In the **Properties > Description** view, change the default **Correlation Set** name to: **CustomerApplication**
- ___ 2. Define new correlation properties for the **AccountVerification** business process.
- ___ a. In **Correlation Properties** in the right pane, click **Add a Correlation Property(+)** icon.
- ___ b. In **Add a Correlation Property**, change the correlation property **Name** to: **CompanyName**
- ___ c. From **Matching data types**, select **string**.
- ___ d. Click **OK**.
- ___ e. For **CompanyName**, double-click the **Properties** tab to enlarge it to full screen.
- ___ f. In the list, find the **AccountVerification** interface.
- ___ g. Underneath it, click the **InputCriterion** operation.
- ___ h. Click **Add**.
- ___ i. At the **Create Property Alias** prompt, expand the nodes until you reach the **companyName:string** property.

- __ j. Leave **Direction** set to **Input**.



- __ k. Click **OK**.
__ l. In the list, remain at the **AccountVerification** interface.
__ m. Underneath **AccountVerification**, click the **InputCriterion** operation again.
__ n. Click **Add**.
__ o. Set **Direction** to **Output**.

- ___ p. At the **Create Property Alias** prompt, expand the nodes until you reach the **message:string** property.



- ___ q. Click **OK**.
- ___ r. In the list, find the **CustomerTypeInterface** interface.
- ___ s. Underneath **CustomerTypeInterface**, click the **customerType** operation.
- ___ t. Click **Add**.
- ___ u. At the **Create Property Alias** prompt, expand the nodes until you reach the **companyName:string** property.
- ___ v. Leave **Direction** set to **Input**.
- ___ w. Click **OK**.
- ___ x. In the list, again find the **CustomerTypeInterface** interface.
- ___ y. Underneath **CustomerTypeInterface**, click the **customerType** operation.
- ___ z. Click **Add**.
- ___ aa. In **Direction**, click **Output**.
- ___ ab. At the **Create Property Alias** prompt, expand the nodes until you reach the **companyName:string** property.

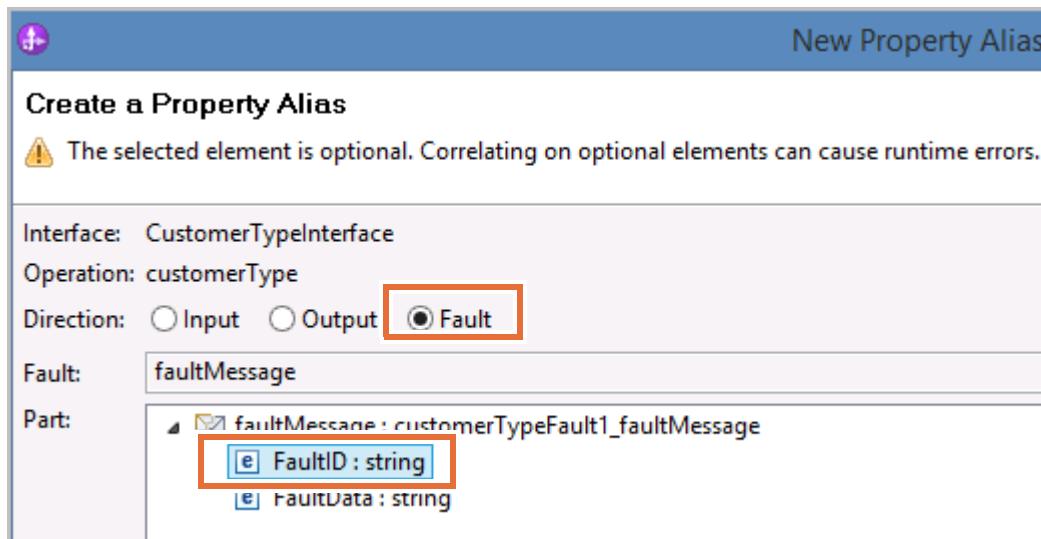
- __ ac. Click **OK**. You now have two entries that are defined:

The screenshot shows two tables representing WSDL operations:

AccountVerification	
InputCriterion	InputCriterionParameters /Input/companyName
	InputCriterionResult /Output/message

CustomerTypeInterface	
customerType	customerTypeParameters /inputCustomerApplication/companyName
	customerTypeResult /outputCustomerApplication/companyName

- __ ad. In the list, again find the **CustomerTypeInterface** interface.
 __ ae. Underneath it, click the **customerType** operation.
 __ af. Click **Add**.
 __ ag. In **Direction**, click **Fault**.
 __ ah. At the **Part** prompt, expand the nodes until you reach the **FaultID:string** property.



- __ ai. Click **OK**.

The screenshot shows the **CustomerTypeInterface** table with an additional row:

CustomerTypeInterface	
customerType	customerTypeParameters /inputCustomerApplication/companyName
	customerTypeResult /outputCustomerApplication/companyName
	faultMessage /FaultID



Information

It is important for you to understand the purpose of correlation sets. Notice that both the **InputCriterion** operation of **AccountVerification** and the **CustomerTypeInterface** interface define request/response operations. Using correlation sets, you are correlating these outgoing and incoming messages with an instance at run time. Thus, if a fault occurs when a request of customer John Doe is processing, the fault must be caught inside the fault handler of the particular process instance that is handling the request of John. Notice that the **FaultID** is also a part of **CompanyName** correlation set for this reason.

- ___ aj. Double-click the **Properties** tab again to resize your IBM Integration Designer panes.
- ___ ak. Save your changes.
- ___ 3. Associate the **Correlation Properties** with the **Correlation Set**.
 - ___ a. Save your work. After saving, you notice that the **CustomerApplication** correlation set is marked as containing a problem. Hovering over it produces this message: **The correlation set must refer to at least one property (correlation set 'CustomerApplication')**.
 - ___ b. Click the **CustomerApplication** correlation set.
 - ___ c. In the **Properties** view, click the check box next to **CompanyName**.

Correlation Set - CustomerApplication		
Description	Name:*	CustomerApplication
	Select correlation properties to identify this process instance. You can create corr...	
	3 Property name	Type
	<input checked="" type="checkbox"/> CompanyName	string

- ___ d. Save your changes.
- ___ 4. Implement the extended customer correlation set in the **AccountVerification** process.
 - ___ a. In the business process editor, click the **Account Verification Receive** activity.
 - ___ b. Click the **Correlation** tab in the **Properties** view.
 - ___ c. Click **Add** to add a correlation set to the Receive activity.
 - ___ d. Change the **Initiation** field by selecting **Yes** from the list.

Correlation sets:		
Direction	Initiation	Correlation Set
Receive	Yes	CustomerApplication



Information

The execution of this **Receive** activity creates an instance of **AccountVerification** at run time. Therefore, it is the first activity to receive the actual **companyName** value at run time. By setting the **Initiation** to **Yes**, the runtime engine can ensure that the messages that are getting in and out of this particular process instance are associated with a particular **companyName**.

- e. In the business process editor, click the **Customer Type Invoke** activity.
- f. Click the **Correlation** tab in the **Properties** view.
- g. Click **Add** to add a correlation set to the type invoke activity.
- h. Change the **Direction** setting by selecting **Both** from the list. Leave the **Initiation** set to the default value **No** and **Correlation Set** as **CustomerApplication**.

Correlation sets:		
Direction	Initiation	Correlation Set
Both	▼ No	CustomerApplication

- i. Save your changes.

Part 9: Updating the FoundationModule assembly diagram

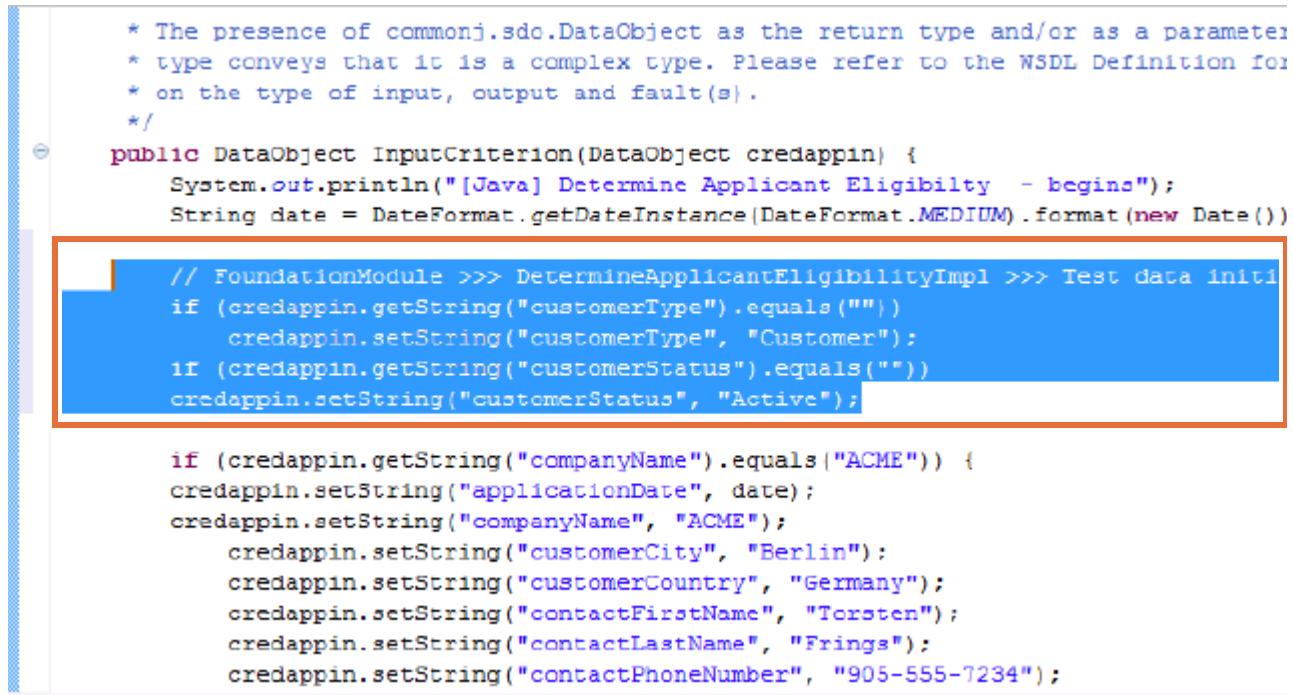
In this part of the exercise, you modify the FoundationModule assembly diagram. You reflect the changes that are made in the **AccountVerification** process in the assembly editor, since the references are changed. You also add the new **CustomerType** SCA component.

- 1. Synchronize the **AccountVerification** implementation with the assembly diagram.
 - a. In the Business Integration view, expand the **FoundationModule** project.
 - b. Double-click **Assembly Diagram** to open it in the assembly editor.
 - c. In the assembly editor, right-click the **AccountVerification** SCA component and select **Synchronize Interfaces and References > from Implementation**.
 - d. Click **Yes** to **Confirm Synchronize from Implementation**.
 - e. In the **Business Integration** view, expand the **FoundationModule > Integration Logic > BPEL Processes > CustomerType**.
 - f. Drag the **CustomerType** process onto the **FoundationModule** assembly diagram.
 - g. Wire the **CustomerTypePartner** reference to the **CustomerType** component by right-clicking **CustomerType** and choosing **Wire to Existing** from the menu.
 - h. Save your changes.
 - i. Check the **Problems** tab. You might receive **Warnings**, but no **Errors** must be present now.

Part 10: Creating test data to test fault handler

- 1. Set up test data creation. Test data is created in the **DetermineApplicationEligibility** component. It must be enhanced to generate test data for the **customerType** and **customerStatus** variables.
 - a. In the **Business Integration** view, expand **FoundationModule > Integration Logic > Java Usages**.
 - b. Double-click **DetermineApplicantEligibilityImpl**.
 - c. In the code editor, position the cursor on the blank line before the first **if** statement in the **InputCriterion** method (line 39).
 - d. Enter the following Java code or copy and paste the `FoundationModule >>> DetermineApplicantEligibilityImpl >>> Test data initialization code from C:\labfiles\Support Files\EX4\EX4_Java_Snippets.txt.`

```
if (credappin.getString("customerType").equals(""))
    credappin.setString("customerType", "Customer");
if (credappin.getString("customerStatus").equals(""))
    credappin.setString("customerStatus", "Active");
```
 - e. Verify that the code copied to the right location.



```

/*
 * The presence of commonj.sdo.DataObject as the return type and/or as a parameter
 * type conveys that it is a complex type. Please refer to the WSDL Definition for
 * on the type of input, output and fault(s).
 */
public DataObject InputCriterion(DataObject credappin) {
    System.out.println("[Java] Determine Applicant Eligibility - begins");
    String date = DateFormat.getDateInstance(DateFormat.MEDIUM).format(new Date())

    // FoundationModule >>> DetermineApplicantEligibilityImpl >>> Test data initilization
    if (credappin.getString("customerType").equals(""))
        credappin.setString("customerType", "Customer");
    if (credappin.getString("customerStatus").equals(""))
        credappin.setString("customerStatus", "Active");

    if (credappin.getString("companyName").equals("ACME")) {
        credappin.setString("applicationDate", date);
        credappin.setString("companyName", "ACME");
        credappin.setString("customerCity", "Berlin");
        credappin.setString("customerCountry", "Germany");
        credappin.setString("contactFirstName", "Torsten");
        credappin.setString("contactLastName", "Frings");
        credappin.setString("contactPhoneNumber", "905-555-7234");
    }
}
```

- f. Save your changes and close the code tab.

Part 11: Testing fault handler in the AccountVerification process

In this part of the exercise, you use IBM Integration Designer integrated test environment to deploy and test your business processes. The integrated test environment is an IBM Process Server environment for you to test your applications in a local environment.

- 1. Start IBM Process Server.
 - a. In the **Servers** view, right-click **IBM Process Server v8.5.7 at localhost** and choose **Start** from the menu.
 - b. Wait for the server to complete the start procedure.



Note

If the server start status shows for more than 5 minutes and the **Server Logs** view is not getting updated with server start messages, then you need to start it manually. Right-click **IBM Process Server v8.5.7 at localhost** and select **Stop** from the menu to stop successfully (do the Stop operation one more time if the server status does not show as Stopped). Restart the server by right-clicking **IBM Process Server v8.5.7 at localhost** and selecting **Start**.

- 2. Publish all projects onto the server for testing.
 - a. Right-click **IBM Process Server v8.5.7** and choose **Add and Remove** from the menu.
 - b. Click **Add All** and click **Finish**. Wait until all modules are published and started.
- 3. Configure the **FoundationModule_Test** test component.
 - a. In the assembly diagram for **FoundationModule**, right-click the **AccountVerification** SCA component and choose **Test Component** from the menu.
 - b. The **FoundationModule_Test** test module tab opens. In the test module editor, select the **Configurations** tab at the bottom of the **FoundationModule_Test** editor.

- ___ c. Note the list of **Monitors**. Presently, no **Emulators** are defined because they are not needed for this test.

Integration Test Client: FoundationModule_Test

Configurations

This area displays test configurations and their resources. Select a test configuration or resource to display its properties in the General Properties and Detailed Properties sections. [More...](#)

```

Test configuration Default Module Test
  - Module FoundationModule
    - Component and Reference Emulators
    - Human Task Emulators
  - Monitors
    - AccountVerification.DetermineApplicationEligibilityPartner -> DetermineApplicantEligibility
    - AccountVerification.RecordIneligibleApplicationPartner -> RecordIneligibleApplication
    - AccountVerification.CreditCheckServicePartner -> CreditScoreServiceImport
    - AccountVerification.CreditRiskAssessmentPartner -> CreditRiskAssessmentImport
    - AccountVerification.RequestMoreDocumentationPartner -> RequestMoreDocumentation
    - AccountVerification.FinalApplicationReviewPartner -> FinalApplicationReview
    - AccountVerification.GenerateDeclinePartner -> GenerateDecline
    - AccountVerification.RecordDeclinedApplicationPartner -> RecordDeclinedApplication
    - AccountVerification.MapToIneligiblePartner -> MapToIneligible
    - AccountVerification.CustomerTypePartner -> CustomerType
    - CreateApplication.AccountVerification -> AccountVerification
    - AccountVerificationExport.<export> -> AccountVerification
  - Fine-Grained Traces
    - AccountVerification (Process:AccountVerification)
    - CustomerType (Process:CustomerType)

```

Events **Configurations**

- ___ 4. Test the **AccountVerification** and **CustomerType** processes by using the component test.
- ___ a. Select the **Events** tab in the **FoundationModule_Test** test component.
- ___ b. In the **Initial request parameters** section, set the **companyName** attribute value to: **IBM**
- ___ c. Set the **customerType** attribute value to: **Competitor**
- ___ d. Set the **customerStatus** attribute value to: **Active**
- ___ e. Click **Continue** to trigger a new **InputCriterion** operation test invocation.
- ___ f. When prompted for the **Deployment Location**, leave **IBM Process Server v8.5.7** as selected and click **Finish**.
- ___ g. When prompted for **User Login-Default Module Test**, accept the default of **admin** for **user ID** and **web1sphere** for **Password**, and click **OK**.
- ___ 5. Review the **Events** that are associated with this business process execution.
- ___ a. Wait for the test execution to complete.

- ___ b. Click through the **Events** in the left pane. Clicking **Invoke** displays the **Invocation parameters** data that initiated the business process.
- ___ c. The **Request and Response for AccountVerification > DetermineApplicantEligibility:InputCriterion** show the **Request parameters** and **Response parameters**. Notice that the parameters are populated completely during this service call.

Name	Type	Value
applicationDeci	boolean	true
comments	string	None
companyName	string	IBM
contactFirstNam	string	Landon
contactLastNam	string	Donovan
contactPhoneN	string	547-555-3172
creditRating	string	A++
creditReportNe	boolean	true
creditRisk	string	LOW
creditScore	int	0

- ___ d. Next, the **Request for CustomerType:customerType** is made. Since **customerType** is **Competitor**, a fault is generated.

- e. View the **Response** for **CustomerType:customerType**. It returned **Fault data**, which can be viewed here, namely, a **faultMessage**. The **FaultID** indicates **IBM**, the primary key of the data that led to this fault. **FaultData** is **Competitor**, the value that created this fault.

The screenshot shows the IBM Worklight Studio interface. On the left, the 'Events' panel displays a trace of events for a test trace. One event, 'Response (AccountVerification <-- C)', is highlighted with a red box. This event is associated with an exception, 'Exception (faultMessage)'. On the right, detailed properties for this event are shown. The 'Module' is 'FoundationModule', 'Source component' is 'AccountVerification', 'Source reference' is 'CustomerTypePartner', 'Target component' is 'CustomerType', 'Target interface' is 'CustomerTypeInterface', and 'Target operation' is 'customerType'. Under 'Fault data', there is a table with three rows:

Name	Type	Value
faultMessage	FaultMessage	IBM
FaultID	string	IBM
FaultData	string	Competitor

- __ f. Finally an exception is shown with its **faultMessage** parameters.

Integration Test Client: FoundationModule_Test

Events

This area displays the events in a test trace. Select an event to display its properties in the General Properties and Detailed Properties sections. [More...](#)

The screenshot shows the 'Events' tab of the Integration Test Client. The list of events includes:

- Customer Type Invoke (customerType)
- Fault Snippet
- Rethrow
- Default Snippet
- Assign Fault
- Reply Fault
- Terminate1
- Request (AccountVerification --> DetermineApplicantEligibility:InputCriterion)
- Response (AccountVerification <- DetermineApplicantEligibility:InputCriterion)
- Request (AccountVerification --> CustomerType:customerType)
- Fine-Grained Trace (CustomerType:CustomerType)
 - Receive (customerType)
 - Customer Application Assign
 - Customer Type Validation
 - Customer Type Choice
 - Assign Fault
 - Reply Fault
- Response (AccountVerification <- CustomerType:customerType)
- Exception (faultMessage) Exception (faultMessage)
- Invoke returned

- __ 6. View the output messages.

- __ a. Click the **Server Logs** tab and view the messages.

[Java] Determine Applicant Eligibility - ends
>>> CustomerType >>> Customer Type Snippet: IBM007 - IBM - Competitor - false
>>> AccountVerification >>> Type Validation Scope >>> Catch faultMessage: IBM - Competitor
>>> AccountVerification >>> Default Fault Handler >>> Catch faultMessage: IBM - Competitor

- __ b. The sequence of the messages must be:

```
>>> CustomerType >>> Customer Type Snippet: IBM007 - IBM - Competitor - false
>>> AccountVerification >>> Type Validation Scope >>> Catch faultMessage: IBM - Competitor
>>> AccountVerification >>> Default Fault Handler >>> Catch faultMessage: IBM - Competitor
```

- __ c. The business process is initiated through the test tool (beginning with **Input Snippet**).

- ___ d. **DetermineApplicantEligibility** populates the rest of the **CustomerApplication** business object with test data that results from a lookup for the primary key, **IBM**.
 - ___ e. **Customer Type Snippet** shows the account number, **IBM007**, the company name, **IBM**, and the customer type, **Competitor**. The last data value results in **false** because this data is not the defined **Customer** value. It triggers an exception that is thrown in the **Type Validation Scope**, and caught in its exception handler. The exception is rethrown to the **Default Fault Handler** and the business process terminates.
- ___ 7. Remove the applications from the server.
 - ___ a. Right-click **IBM Process Server v8.5.7** and click **Add and Remove** from the menu.
 - ___ b. Click **Remove All** and click **Finish**.
 - ___ c. Close all the open tabs. Do not save the **FoundationModule_Test**.
 - ___ 8. (Optional) Stop IBM Process Server.
 - ___ a. To stop IBM Process Server, in the **Servers** view, right-click **IBM Process Server v8.5.7 at localhost**.
 - ___ b. Choose **Stop** from the menu.
 - ___ 9. Do not close IBM Integration Designer.

End of exercise

Exercise 5. Applying a compensation handler to WS-BPEL

Estimated time

01:00

Overview

This exercise shows you how to implement compensation handlers in a business process. Since you cannot predict exactly when a particular fault might happen, you need to define a sequence of activities to react to such events, and possibly undo the activities that are already committed. The compensation handlers can handle these types of scenarios.

Objectives

After completing this exercise, you should be able to:

- Implement a compensation handler in a business process
- Test compensation handlers in the IBM Integration Designer integrated test environment

Exercise instructions

Preface

The following business objects are used in this exercise.

- **CustomerApplication:** The simple (non-hierarchical) business object that describes the characteristics of a particular customer application record. This generic business object (GBO) is instantiated from the user input through the human task.

The following interfaces are used along with the **CustomerApplication** business object.

- **CustomerApplicationDelivery:** Contains a single one-way operation that passes a single part, named **inputCustomerApplication** of type **CustomerApplication**, on the request. This interface is used on the definition of the business process and is exposed as part of the exported SCA binding, which the other SCA components call.
- **CustomerApplicationRequest:** Contains a single request/response operation that passes a single part, named **inputCustomerApplication** of type **CustomerApplication**, on the request and a single part, named **outputCustomerApplication** of type **CustomerApplication**, on the response. With the use of this interface, it is expected that any changes that are made to **CustomerApplication** on the receiving end are passed back to the process. This interface is used to define reference partners in the business process. Using the same interface simplifies the data exchange between the processes.

Compensation handling

Part 1: Extending the default fault handler in the AccountVerification process

The default top-level fault handler catches all faults that get propagated up to the process level scope. You then implement a fault-specific catch that processes an explicit fault type. Next, you add logic to trigger a compensation.



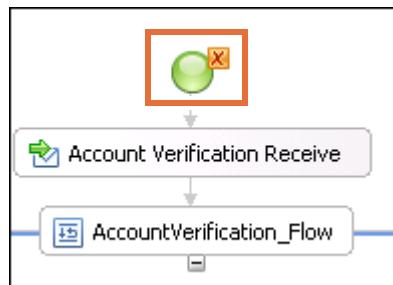
Note

After you complete this exercise, the complete BPEL process diagram must be as shown in the diagram in the `FoundationModule_bpel_AccountVerification.jpg` file at the `C:\labfiles\Support Files\EX5` folder. You can refer to this diagram as you progress in this exercise to validate your work.

- ___ 1. Open the Exercise 5 workspace.
 - ___ a. On your desktop, open the folder that is labeled **Exercise Shortcuts**.
 - ___ b. Double-click the shortcut that is labeled **Exercise 5** to start IBM Integration Designer.
 - ___ c. Wait for the workspace build to complete; it might take a few minutes.
 - ___ d. Close the **Getting Started** tab.
- ___ 2. Add a new compensate activity to the **Catch1** catch construct.

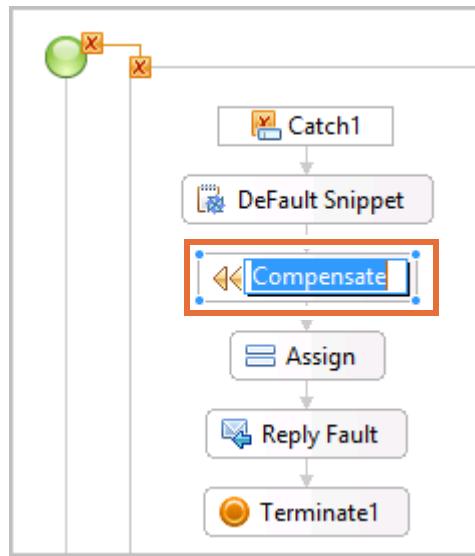
In a long-running process, each transaction is committed individually. If the process fails, a compensate activity specifies how to deal with each transaction that needs to be compensated. The compensation handler of a scope, or a process, triggers the compensation.

 - ___ a. In the **Business Integration** view, expand **FoundationModule > Integration Logic > BPEL Processes > AccountVerification**.
 - ___ b. Double-click the **AccountVerification** process to open it in the business process editor.
 - ___ c. In the business process editor, click the start node (a green circle) to open a top-level fault handler, unless it is already open.

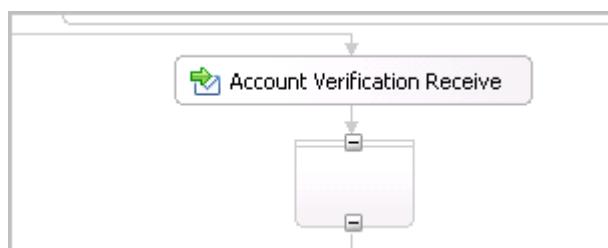


- ___ d. In the business process editor palette, in the **Faults** drawer, click the **Compensate** icon.

- __ e. Click below the **DeFault Snippet** of the **Catch1** catch construct to add a new **Compensate** activity.



- __ f. Accept the default display name.
- __ 3. Define a new receive scope inside the **AccountVerification** BPEL process to enclose a portion of the logic that is related to the business process initialization.
- __ a. In the business process editor palette, in the **Structures** drawer, click the **Scope** icon.
 - __ b. Click below the **Account Verification Receive** activity to add a **Scope** activity.
 - __ c. Click the new **Scope** activity.
 - __ d. Select the **Description** tab in the **Properties** view.
 - __ e. In the **Display name** field, enter: **Receive Scope**

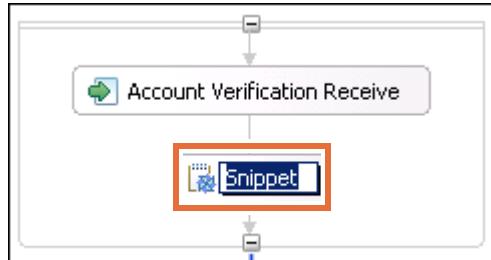


- __ f. Drag **Account Verification Receive** into **Receive Scope**.



- __ g. Save your changes by pressing **Ctrl+S**.
- __ 4. Add an input snippet to the scope.
- __ a. Select **Snippet** under **Basic Actions** from the palette.

- ___ b. Click inside **Receive Scope** below the **Account Verification Receive** activity to add a **Snippet** activity.



- ___ c. Click the new **Snippet** activity.
- ___ d. Select the **Description** tab in the **Properties** view.
- ___ e. In the **Display name** field, enter: Input Snippet
- ___ f. In the **Properties** view for the new **Input Snippet** activity, click the **Details** tab.
- ___ g. Select the **Java** option to indicate that the snippet logic is going to be defined as Java code. If a Question window pops up, select **Yes**.
- ___ h. Enter the following Java code in the window or copy the `AccountVerification >>> Input Snippet` to `Input Snippet` from `C:\labfiles\Support Files\EX5\EX5_Java_Snippets.txt`. You do not need to copy the comment lines that begin with `//`.

```
System.out.println(">>> AccountVerification >>> Input Snippet >>>
CustomerApplicationVariable:" +
"companyName: " + CustomerApplicationVariable.getString("companyName"));
```

- ___ i. Save your changes.
- ___ 5. Add a compensation handler to the **Receive Scope** construct.
- ___ a. Right-click the **Receive Scope** activity.
- ___ b. Choose **Add a Compensation Handler** from the menu.
- ___ c. In the business process editor palette, select the **Snippet** icon.
- ___ d. Click inside the newly created **Compensation Handler** to add the **Snippet** activity.

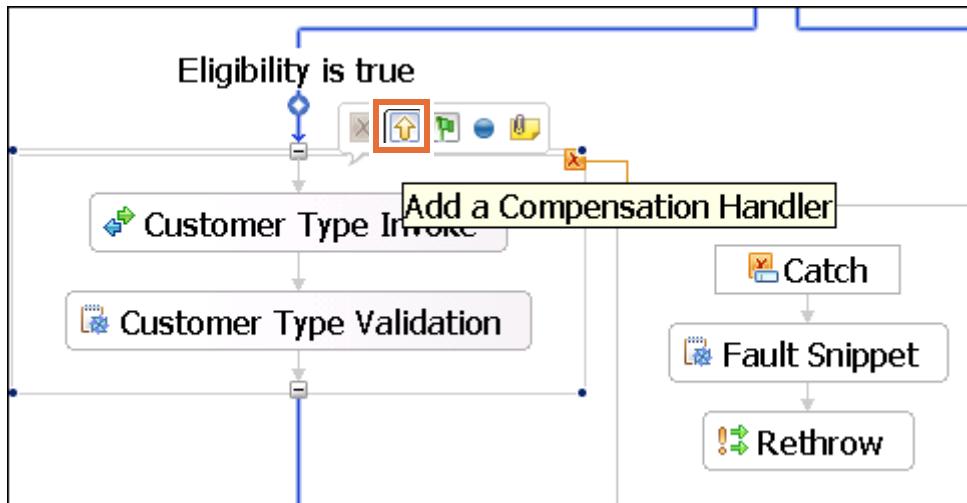


- ___ e. Select the new snippet activity and change the display name to: **Receive Scope Snippet**
- ___ f. In the **Properties** view, select the **Details** tab.
- ___ g. Select the **Java** option to indicate that the code is going to be entered in Java format. If a Question window pops up, select **Yes**.

- __ h. Enter the following Java code in the window or copy the AccountVerification >>> Compensation Handler >>> Receive Scope Snippet to Receive Scope Snippet from C:\labfiles\Support Files\EX5\EX5_Java_Snippets.txt. You do not need to copy the comment lines that begin with //.

```
System.out.println(">>> AccountVerification >>> Compensation Handler >>>
Receive Scope Snippet");
```

- __ i. Save your changes.
- __ 6. Add a compensation handler for the **Type Validation Scope** activity.
- __ a. Select the **Type Validation Scope** activity and click the **Add a Compensation Handler** icon from the menu.

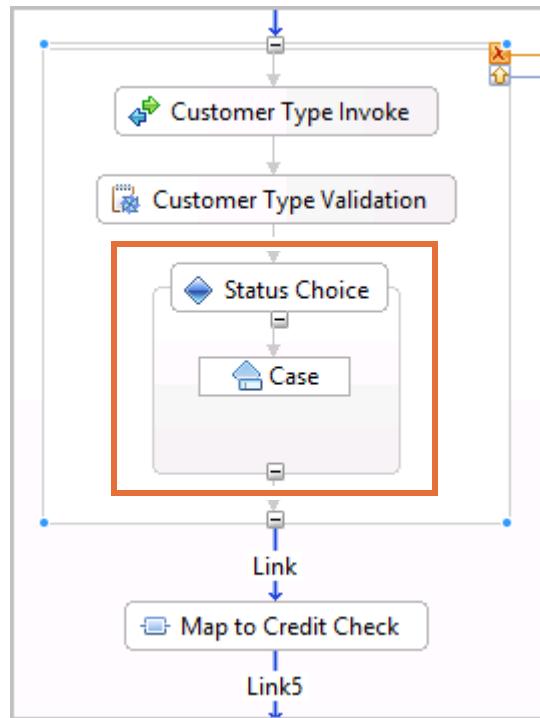


- __ b. In the business process editor palette, select the **Snippet** icon.
- __ c. Click inside the newly created **Compensation Handler** activity to add a snippet.
- __ d. Select the new snippet activity and change the display name to: **Type Validation Scope Snippet**
- __ e. In the **Properties** view, select the **Details** tab.
- __ f. Select the **Java** option to indicate that the code is going to be inserted in Java format. If a Question window pops up, select **Yes**.
- __ g. Enter the following Java code in the window or copy the AccountVerification >>> Compensation Handler >>> Type Validation Scope Snippet to Validation Scope Snippet from C:\labfiles\Support Files\EX5\EX5_Java_Snippets.txt. You do not need to copy the comment lines that begin with //.
- ```
System.out.println(">>> AccountVerification >>> Compensation Handler >>>
Type Validation Scope Snippet");
```
- \_\_ h. Save your changes.

## Part 2: Implementing a new status validation in the AccountVerification process

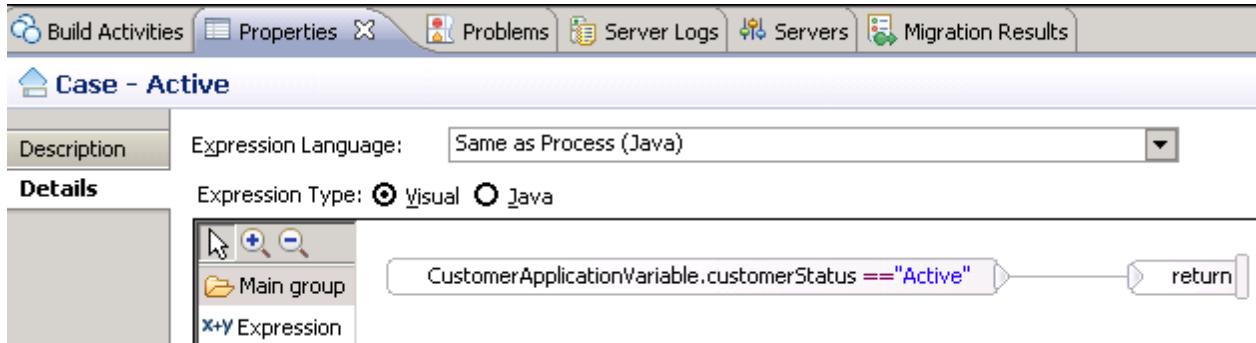
**AccountVerification** is already checking the customer type information through a service call to the **CustomerType** process. You now add logic to the **AccountVerification** process to also check the customer status and raise a fault for a detected inactive customer status setting. The logic checks the status of the customer. If the customer is active, no further action is necessary (other than a status message to the server logs or console). For inactive customer status, a fault is thrown.

- \_\_\_ 1. Add a **Choice** construct between **Customer Type Validation** and **Map to Credit Check**.
  - \_\_\_ a. In the business process editor palette, click the **Choice** icon in the **Structures** drawer.
  - \_\_\_ b. Click below the **Customer Type Validation** to add a **Choice** construct.
  - \_\_\_ c. For readability, right-click the diagram and choose **Arrange Parallel Activities Contents**.
  - \_\_\_ d. Select the **Properties > Description** tab and change the default **Choice** display name to: Status Choice



- \_\_\_ e. Select the **Case** construct, and on the **Description** tab in the **Properties** view, enter the display name: Active
- \_\_\_ f. In the **Properties** view, select the **Details** tab for the **Active** case construct.
- \_\_\_ g. Expand the **Expression language** list and select **Same as Process (Java)**. A visual snippet editor opens automatically.
- \_\_\_ h. Click inside the **true** snippet to expand the expression builder list.
- \_\_\_ i. Expand **CustomerApplicationVariable** and select the **customerStatus:nillable:string** attribute.

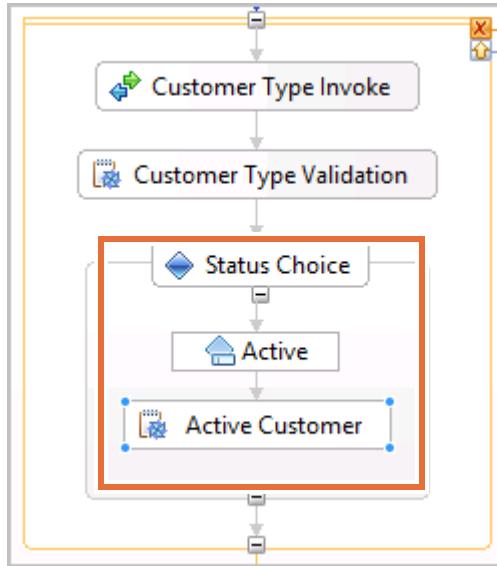
- \_\_ j. Click at the end of the **CustomerApplicationVariable.customerStatus** expression, and select **==** (double-equals) from the expression builder list.
- \_\_ k. Click at the end of the **CustomerApplicationVariable.customerStatus ==** expression, scroll down in the list, and select **String**.
- \_\_ l. In the **Type a string** box, enter **Active** into the input field. Do not use quotation marks; the editor inserts them for you automatically.
- \_\_ m. Press the **Enter** key. The snippet must look like the following screen capture:



- \_\_ n. Save your changes.
2. Complete the Status Choice **Active** case.
    - \_\_ a. In the **Palette**, click **Snippet**.
    - \_\_ b. Place the snippet in the **Status Choice** below the **Active** case.
    - \_\_ c. Change the snippet display name to: **Active Customer**
    - \_\_ d. In the **Properties** view for the **Active Customer** snippet activity, click the **Details** tab.
    - \_\_ e. Select the **Java** option to indicate that the snippet logic is going to be defined as Java code. If a Question window pops up, select **Yes**.
    - \_\_ f. Enter the following Java code in the window or copy the **AccountVerification >>> Active Customer Snippet** to Active Customer Snippet from **C:\labfiles\Support Files\EX5\EX5\_Java\_Snippets.txt**. You do not need to copy the comment lines that begin with **//**.

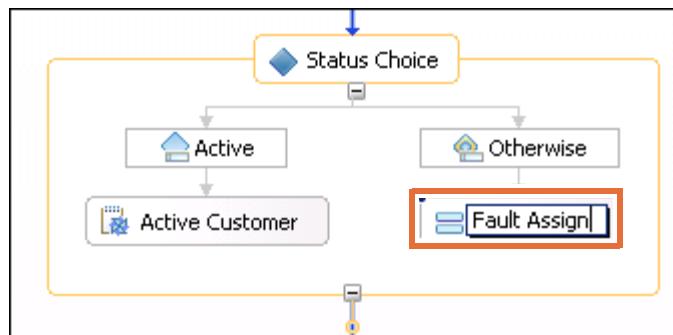
```
System.out.println(">>> AccountVerification >>> Active Customer Snippet:
" +
CustomerApplicationVariable.getString("accountNumber") + " - " +
CustomerApplicationVariable.getString("companyName") + " - " +
CustomerApplicationVariable.getString("customerType"));
```

- \_\_ g. Save your changes.



- \_\_ 3. Implement the **Otherwise** case for **Status Choice**.

- \_\_ a. Select **Status Choice**, and click the **Add an Otherwise Element** icon from the menu.
- \_\_ b. In the business process editor palette, click the **Assign** icon.
- \_\_ c. Click under the **Otherwise** icon to add the **Assign** activity.
- \_\_ d. Change the display name to: **Fault Assign**

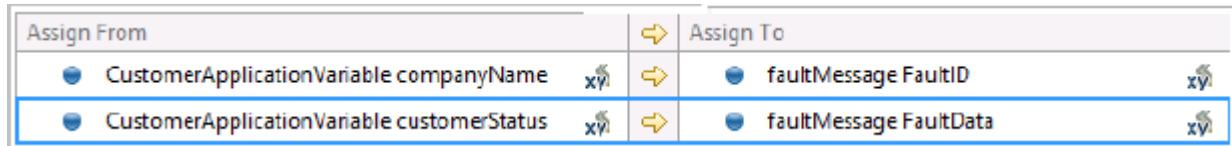


- \_\_ e. In the **Properties** view for **Fault Assign**, select the **Details** tab.
- \_\_ f. Click **Select From**.
- \_\_ g. Select **CustomerApplicationVariable > companyName: string**.
- \_\_ h. For **Select To**, select the **faultMessage: Fault Message > FaultID: string** attribute.

| <b>Assign - Fault Assign</b> |                                         |           |
|------------------------------|-----------------------------------------|-----------|
| Description                  | Assign From                             | Assign To |
| <b>Details</b>               | CustomerApplicationVariable companyName |           |
| Server                       |                                         |           |

- \_\_ i. Click **Add** to create a second assignment.

- \_\_\_ j. Click **Select From** and select **CustomerApplicationVariable > customerStatus: string**.
- \_\_\_ k. For **Select To**, select the **faultMessage: FaultMessage > FaultData: string** attribute.



- \_\_\_ 4. Add a snippet to the **Otherwise** construct to print the fault message.
  - \_\_\_ a. In the business process editor, add a **Snippet** activity under the **Fault Assign**.
  - \_\_\_ b. For readability, right-click the diagram and choose **Arrange Parallel Activities Contents**.
  - \_\_\_ c. Select the new **Snippet** and change the display name to: **Inactive Customer**
  - \_\_\_ d. In the **Properties** view, select the **Details** tab.
  - \_\_\_ e. Select the **Java** option to indicate that the code is entered in Java format. If a Question window pops up, select **Yes**.
  - \_\_\_ f. Enter the following Java code in the window or copy **AccountVerification >>> Status Choice >>> Otherwise - faultMessage** to **Inactive Customer Snippet** from **C:\labfiles\Support Files\EX5\EX5\_Java\_Snippets.txt**. You do not need to copy the comment lines that begin with **//**.
 

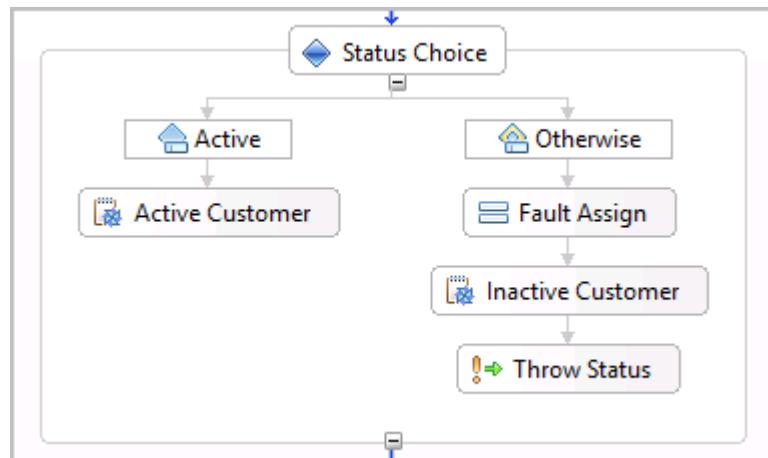
```
System.out.println("">>>> AccountVerification >>> Status Choice >>>
Otherwise - faultMessage: " +
faultMessage.getString("FaultID") + " - " +
faultMessage.getString("FaultData"));
```
  - \_\_\_ g. Save your changes.
- \_\_\_ 5. Now that the **AccountVerification** can detect an invalid customer status, you add a new throw activity to the **Status Choice – Otherwise** logic to throw a fault message.
  - \_\_\_ a. In the business process editor, click a **Throw** activity in the **Faults** drawer and then click it below the newly created **Inactive Customer** activity.
  - \_\_\_ b. For readability, right-click the diagram and choose **Arrange Parallel Activities Contents**.
  - \_\_\_ c. In the **Properties** view, click the **Description** tab.
  - \_\_\_ d. Change the **Display name** of throw to: **Throw Status**
  - \_\_\_ e. In the **Properties** view, click the **Details** tab.
  - \_\_\_ f. In the **Fault Name** field, type: **faultMessage**
  - \_\_\_ g. In the **Fault Variable** field, click **Browse** to look up a fault variable.
  - \_\_\_ h. At the **Select Fault Variable** prompt, select **faultMessage** and click **OK**.

- \_\_ i. In the **Namespace** field, type:

`http://FoundationLibrary/customertype/CustomerTypeInterface`  
This setting must match the top-level faultMessage catch.

|                 |                                                                          |
|-----------------|--------------------------------------------------------------------------|
| Fault Name:*    | <code>faultMessage</code>                                                |
| Namespace:      | <code>http://FoundationLibrary/customertype/CustomerTypeInterface</code> |
| Fault Variable: | <code>faultMessage</code>                                                |

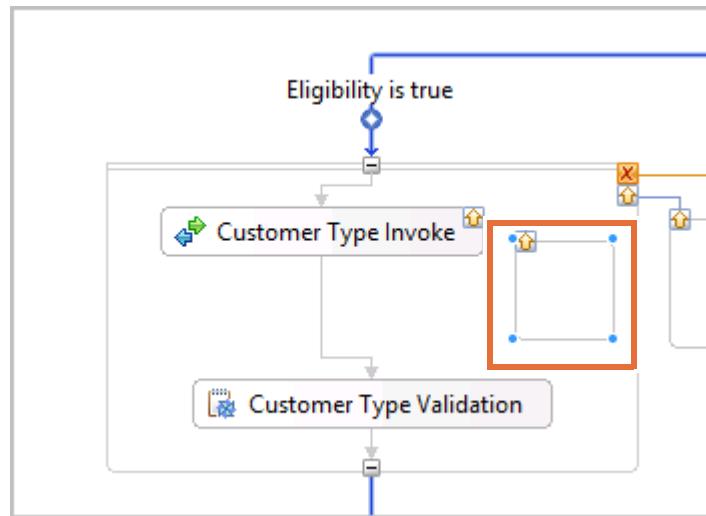
- \_\_ j. For readability, right-click inside the business process editor and choose **Align Parallel Activities Contents Automatically** from the menu.



- \_\_ k. Save your changes.

### Part 3: Adding a compensation handler

- 1. You now add a compensation handler for the **Customer Type Invoke** activity that is nested inside the **Type Validation Scope** construct. The scope and the invoke activity have a compensation handler.
- a. Select the **Customer Type Invoke** activity and select the **Add a Compensation Handler** icon from the menu.



- b. Select a new **Snippet** activity from the palette and drop it inside the new compensation handler.
- c. Change the **Snippet** display name to: **Type Validation Invoke Snippet**
- d. In the **Properties** view, select the **Details** tab.
- e. Select the **Java** option to indicate that the code is inserted in Java format.
- f. Enter the following Java code in the window or copy `AccountVerification >>> Compensation Handler >>> Type Validation Invoke Snippet` from `C:\labfiles\Support Files\EX5\EX5_Java_Snippets.txt`. You do not need to copy the comment lines that begin with `//`.
 

```
System.out.println(">>> AccountVerification >>> Compensation Handler >>> Type Validation Invoke Snippet");
```
- g. Press **Ctrl+S** to save and **Ctrl+W** to close the editor.

### Part 4: Testing compensation handler in the AccountVerification process

In this part of the exercise, you use IBM Integration Designer integrated test environment to deploy and test your business processes. The integrated test environment is an IBM Process Server environment for you to test your applications in a local environment.

- 1. If necessary, start IBM Process Server.
- a. In the **Servers** view, right-click **IBM Process Server v8.5.7 at localhost** and choose **Start** from the menu.
- b. Wait for the server to complete the start procedure.

**Note**

If the server start status shows for more than 5 minutes and the **Server Logs** view is not getting updated with server start messages, then you need to start it manually. Right-click **IBM Process Server v8.5.7 at localhost** and select **Stop** from the menu to stop successfully (do the Stop operation one more time if the server status does not show as Stopped). Restart the server by right-clicking **IBM Process Server v8.5.7 at localhost** and selecting **Start**.

- 2. Publish all projects onto the server for testing.
  - a. Right-click **IBM Process Server v8.5.7 at localhost** and choose **Add and Remove** from the menu.
  - b. Click **Add All** and click **Finish**. Wait until all modules are published and started.
- 3. Configure and test the **AccountVerification** and **CustomerType** processes by using the **FoundationModule\_Test** test component.
  - a. In the assembly diagram for the **FoundationModule**, right-click the **AccountVerification** SCA component and choose **Test Component** from the menu.
  - b. Select the **Events** tab in the **FoundationModule\_Test** test component.
  - c. In the **Initial request parameters** section, set the **companyName** attribute value to: **IBM**
  - d. Set the **customerType** attribute value to: **Competitor**
  - e. Set the **customerStatus** attribute value to: **Active**
  - f. Click **Continue** to trigger a new **InputCriterion** operation test invocation.
  - g. When prompted for the **Deployment Location**, leave **IBM Process Server v8.5.7 at localhost** as selected and click **Finish**.
  - h. When prompted for **User Login-Default Module Test**, accept the default of **admin** for **User ID** and **web1sphere** for **Password**, and click **OK**.
- 4. Review the **Events** that are associated with this business process execution.
  - a. Wait for the test execution to complete.
  - b. Click through the **Events** in the left window. Clicking **Invoke** displays the **Invocation parameters** data that initiated the business process.
  - c. The **Request** and the **Response** for **AccountVerification > DetermineApplicantEligibility:InputCriterion** show the **Request parameters** and **Response parameters**. Notice that the parameters are populated completely during this service call.
  - d. Next, the **Request** for **CustomerType:customerType** is made. Since **customerType** is **Competitor**, a fault is generated.

- \_\_ e. View the **Response** for **CustomerType:customerType**. It returns **Fault data**, which can be viewed here, namely, a **faultMessage**. The **FaultID** indicates **IBM**, the primary key of the data that led to this fault. **FaultData** is **Competitor**, the value that created this fault.
- \_\_ f. Finally, the **Exception** is shown with its **faultMessage** parameters.

**Integration Test Client: FoundationModule\_Test**

**Events**

This area displays the events in a test trace. Select an event to display its properties in the General Pro

```

graph TD
 Root[Invoke started] --> Inv1[Invoke (AccountVerification:InputCriterion)]
 Inv1 --> Tr1[Fine-Grained Trace (AccountVerification:AccountVerification)]
 Tr1 --> Rec1[Receive Scope]
 Rec1 --> Inv2[Account Verification Receive (InputCriterion)]
 Inv2 --> Snip1[Input Snippet]
 Tr1 --> Flow1[AccountVerification_Flow]
 Flow1 --> Elig1[Determine Application Eligibility (InputCriterion)]
 Elig1 --> Val1[Type Validation Scope]
 Flow1 --> Req1[Request (AccountVerification --> DetermineApplicantEligibility:InputCriterion)]
 Flow1 --> Res1[Response (AccountVerification <-- DetermineApplicantEligibility:InputCriterion)]
 Flow1 --> Req2[Request (AccountVerification --> CustomerType:customerType)]
 Tr1 --> Cus1[Fine-Grained Trace (CustomerType:CustomerType)]
 Cus1 --> Rec2[Receive (customerType)]
 Cus1 --> Ass1[Customer Application Assign]
 Cus1 --> Val2[Customer Type Validation]
 Cus1 --> Cho1[Customer Type Choice]
 Cus1 --> Res2[Response (AccountVerification <-- CustomerType:customerType)]
 Res2 --> Fault1[Exception (faultMessage)]
 Fault1 --> Inv3[Invoke returned]

```

- \_\_ 5. View the output messages.
  - \_\_ a. Click the **Server Logs** tab.

- b. Locate the log message to indicate that the fault handler triggers the **compensation handlers**. The sequence of the messages must be:

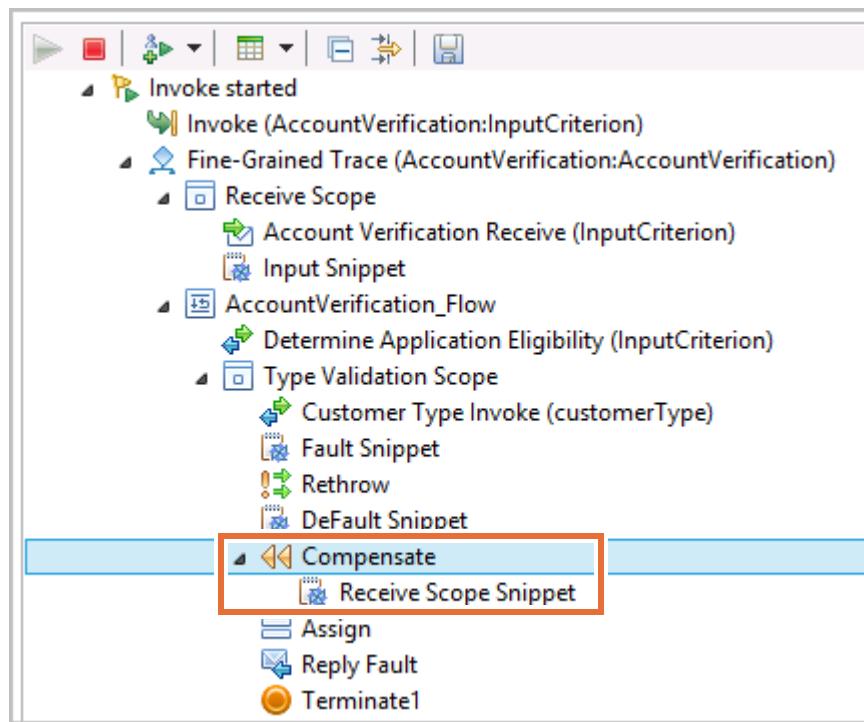
```
>>> CustomerType >>> Customer Type Snippet: IBM007 - IBM - Competitor - false
>>> AccountVerification >>> Type Validation Scope >>> Catch faultMessage: IBM - Competitor
>>> AccountVerification >>> Default Fault Handler >>> Catch faultMessage: IBM - Competitor
>>> AccountVerification >>> Compensation Handler >>> Receive Scope Snippet
```

#### Contents

|                                                                                            |
|--------------------------------------------------------------------------------------------|
| WSVR0220I: Application stopped: FoundationModuleApp                                        |
| WSVR0200I: Starting application: FoundationModuleApp                                       |
| WSVR0221I: Application started: FoundationModuleApp                                        |
| >>> AccountVerification >>> Input Snippet >>> CustomerApplicationVariable:companyName: IBM |
| [Java] Determine Applicant Eligibility - begins                                            |
| [Java] Determine Applicant Eligibility - ends                                              |
| >>> CustomerType >>> Customer Type Snippet: IBM007 - IBM - Competitor - false              |
| >>> AccountVerification >>> Type Validation Scope >>> Catch faultMessage: IBM - Competitor |
| >>> AccountVerification >>> Default Fault Handler >>> Catch faultMessage: IBM - Competitor |
| >>> AccountVerification >>> Compensation Handler >>> Receive Scope Snippet                 |

- c. The business process is initiated through the test tool (beginning with **Input Snippet**).
- d. **DetermineApplicantEligibility** populates the rest of the **CustomerApplication** business object with test data that results from a lookup for the primary key, **IBM**.
- e. **Customer Type Snippet** shows the account number, **IBM007**, the company name, **IBM**, and the customer type, **Competitor**. The last data value results in a value of **false** as it is not the defined **Customer** value.
- f. It triggers an exception that is thrown in the **Type Validation Scope**, and caught in its exception handler.
- g. The exception is rethrown to the default **Fault Handler**.

- h. In the default fault handler, since the compensate activity is specified, it triggers the **Compensation Handler of the Receive Scope**.



- i. Finally, the business process terminates.
- 6. Remove the applications from the server.
- Right-click **IBM Process Server v8.5.7 at localhost**.
  - Choose **Add and Remove** from the menu.
  - Click **Remove All** and click **Finish**.
  - Close all the open tabs. Do not save **FoundationModule\_Test**.
- 7. Close the IBM Integration Designer. Click **File > Exit**, or click **X** in the upper-right corner.

## End of exercise

---

# Exercise 6. Working with business state machines

## Estimated time

01:00

## Overview

This exercise reinforces the business state machine concepts that you learned in the lecture.

## Objectives

After completing this exercise, you should be able to:

- Explore the building blocks of the SodaMachine with the business state machine editor
- Test the SodaMachine state machine with Business Process Choreographer Explorer

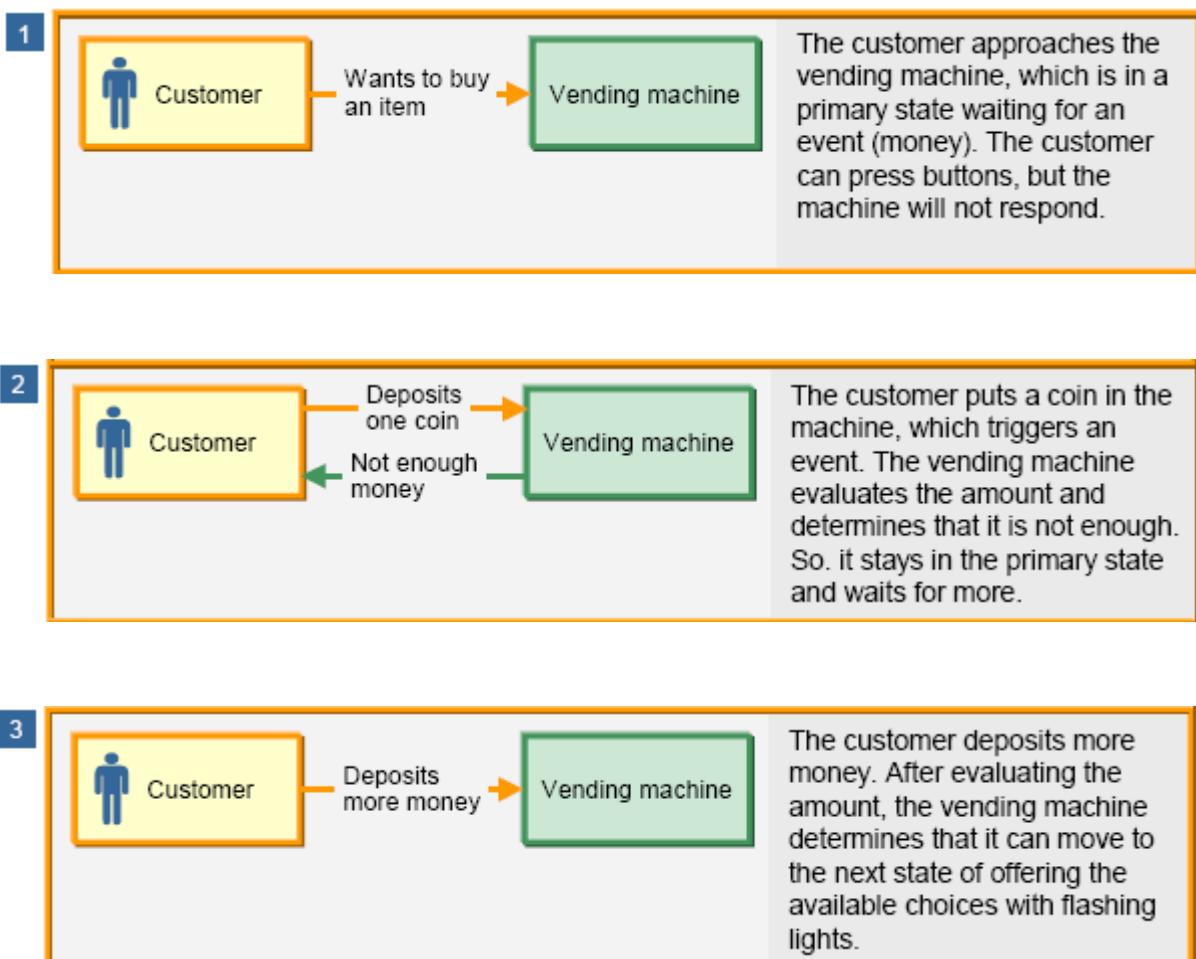
## Exercise instructions

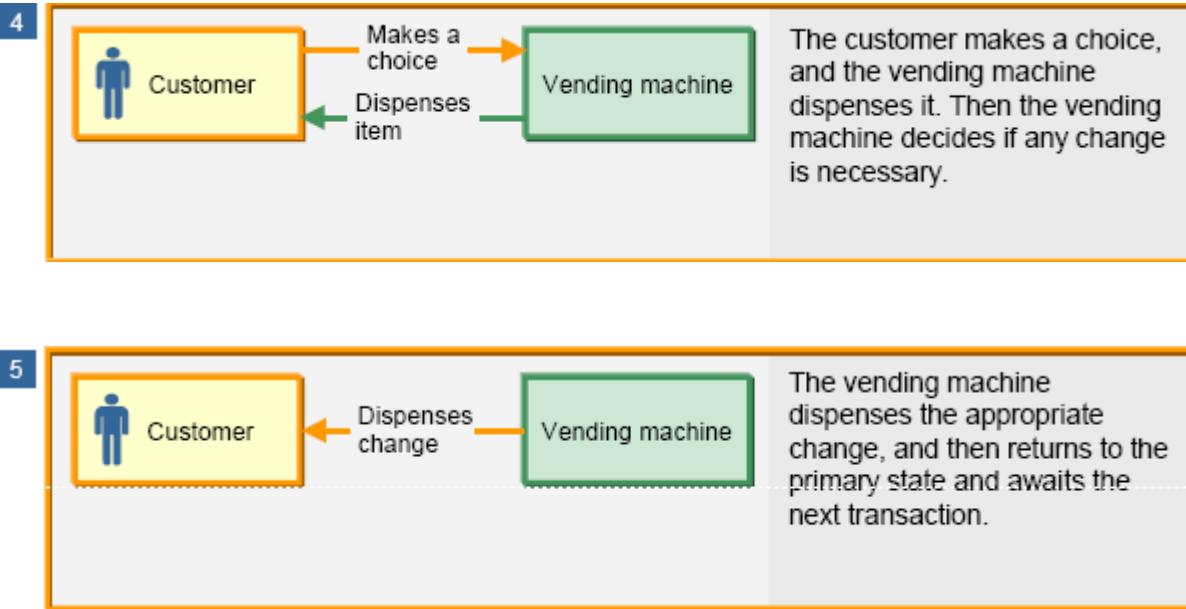
### Introduction to the SodaMachine sample

A business state machine is an event-driven business application in which external operations trigger changes that guide the state machine from one discrete mode to another. Each mode is an individual state, and this mode determines what activities and operations can occur. The state machines are service components, which represent business processes that are based on states and events instead of a sequential business process model.

You create business state machines in IBM Integration Designer by using the business state machine editor.

The SodaMachine sample demonstrates a business state machine that is used to control the purchase of a can of soda from a vending machine. The diagram shows various interactions and activities that are involved in the purchase process.





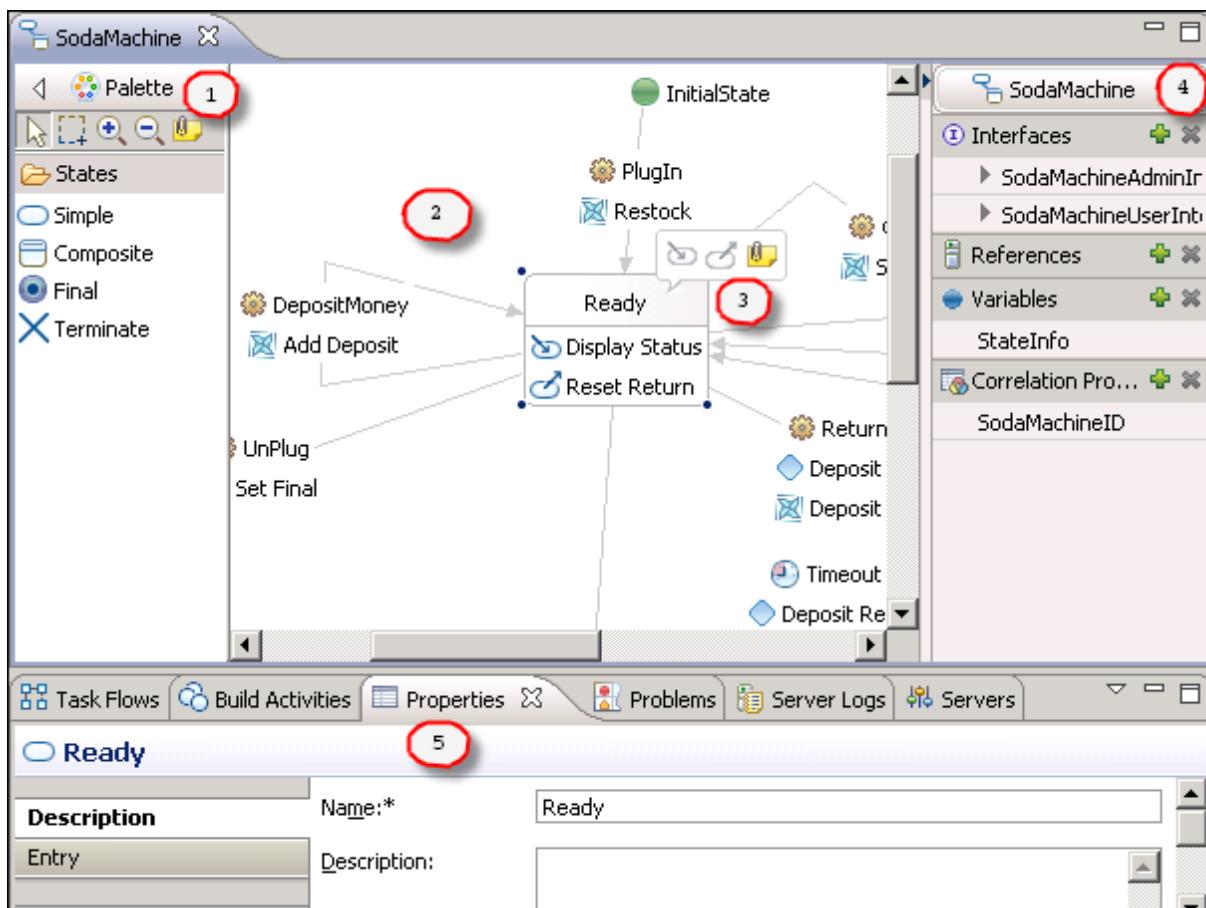
In the various scenarios that are described above, a customer must go through a sequence of interactions to purchase a soda from the vending machine. The vending machine does not respond until the money is deposited. Even then, the vending machine does not proceed to the next state until a certain amount of money is deposited. In each of the states, the actions that the customer can do are unique from any other state. For example, when it is in the primary state, the customer can press product buttons, but doing so has no effect on the transaction. This scenario helps you understand the building blocks of a business state machine.

### ***Part 1: Explore the building blocks of the SodaMachine with the business state machine editor***

In this part of the exercise, you explore the state machine editor and view the various building blocks of the state machine. The business state machine editor is a graphical programming environment that you use to visually create and manipulate business state machines.

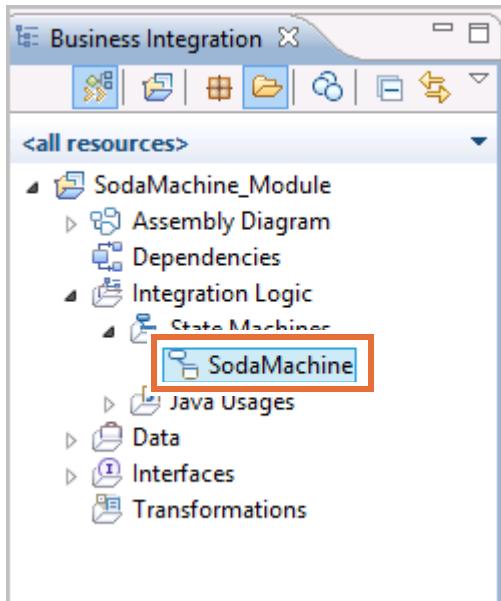
This editor was designed to make you, the user, feel comfortable and to facilitate development. While you are designing your business state machine, think of it as an exercise in how to paint. The white space is called the canvas, and it is where you create your business state machine from the objects that you pull from the palette to the left. Beneath that is an interactive properties area that changes to show pertinent details about whatever object you select on the canvas.

The business state machine editor is divided into several distinct areas, each with its own individual use.

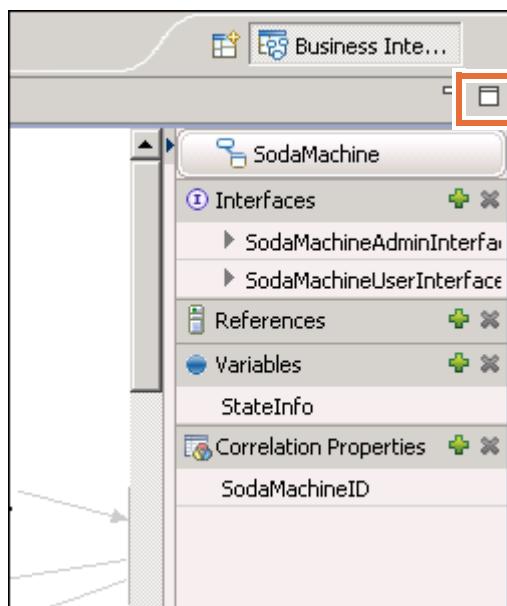


1. **The palette:** The palette is the shaded area to the left of the canvas that houses the states that you click and drag onto the canvas to build your business state machine.
2. **The canvas:** The canvas is the white area in the middle of the editor that you use to assemble the states and transitions to compose your business state machine.
3. **The action bar:** The action bar is a window that pops up beside the object when you select and hover over it, and contains a list of elements that can be added to that object.
4. **The tray:** The tray shows the Interfaces, References, Variables, and Correlation properties that are associated with your business state machine. To create something from the tray, click the corresponding icon.
5. **The properties view:** This view shows properties that are relevant to the object that is selected on the canvas. Click the tabs to the left of this view to toggle through the pages. Some pages show properties in tabular format, and you can add or modify these properties. The contents of the page differ depending on the object chosen. You can press F1 to start a help window for more details.
  - 1. Open the Exercise 6 workspace.
  - a. On your desktop, open the folder that is labeled **Exercise Shortcuts**.
  - b. Double-click the shortcut that is labeled **Exercise 6**.

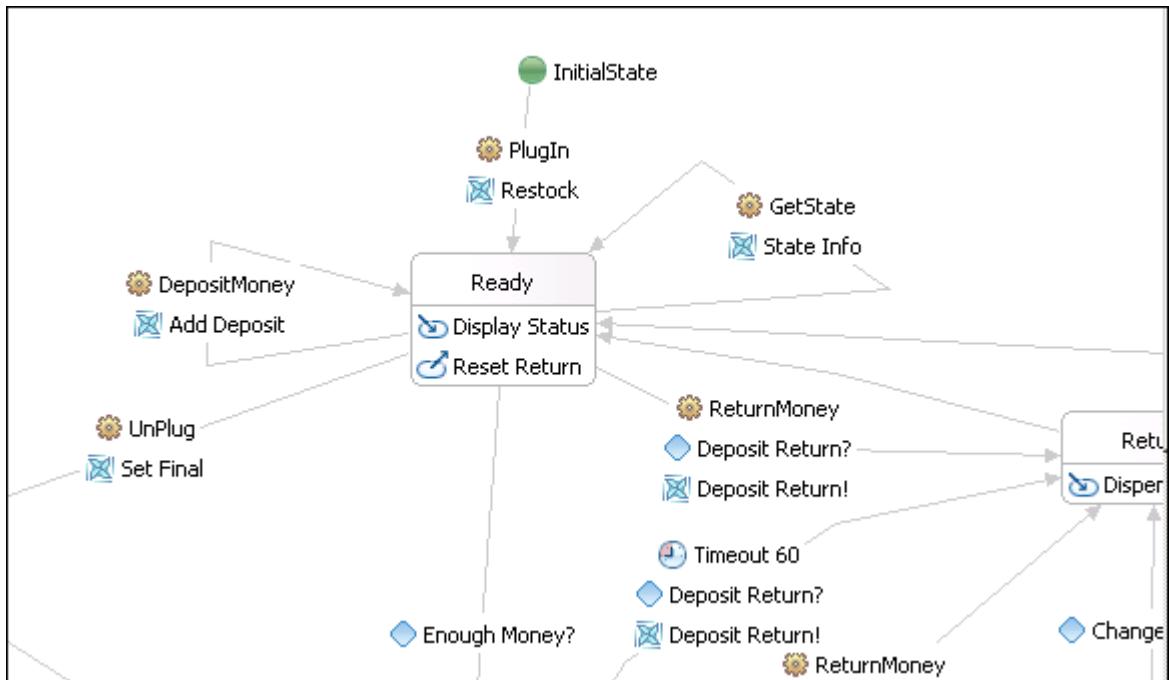
- \_\_\_ c. Close the **Getting Started** tab.
- \_\_\_ 2. Explore the SodaMachine state machine.
  - \_\_\_ a. In the **Business Integration** view, expand **SodaMachine\_Module > Integration logic > State Machines**.
  - \_\_\_ b. Double-click **SodaMachine** to open the **SodaMachine** business state machine editor.



- \_\_\_ c. In the business state machine editor, click the maximize icon on the upper right of the editor to maximize the display.



You can now get a better view of the state machine. You might want to scroll the horizontal and vertical bars to explore the SodaMachine state machine in the editor.



The SodaMachine business state machine is composed of the following building blocks:

- **Interface:** A set of operations to which the state machine responds. In the SodaMachine state machine, the **Interfaces** section lists the operations that the current business state machine advertises. One or more interfaces define the events (operations) that can cause a transition from one state to another.
- **Reference:** Tells the state machine where to find operations that it can start. More specifically, it points to the interface that is used in the invocation of another component. Reference operations can also provide an implementation for *conditions* (guards).
- **Variables:** Store the data that is used within a state machine and can be either a business object or a simple type.
- **Correlation properties:** Are used to distinguish one instance of a state machine from another within a runtime environment.
- **State:** One of several discrete individual stages that represent a business transaction.
- **Transition:** The movement from one state to the next by recognizing an appropriate triggering event. The movement is based on the evaluation of the conditions that are necessary for control to flow through it. During the transition, the associated actions are run. Transition actions, entry actions, and exit actions can start operations that are listed in the References section.
- **Event:** What (external prompt) triggers a transition from one state to another.
- **Condition:** Guards the transition and allows processing when and if it evaluates to true. Otherwise, the current state is maintained.
- **Action:** An activity that is run when a state is entered, exited, or on a transition within a business state machine.

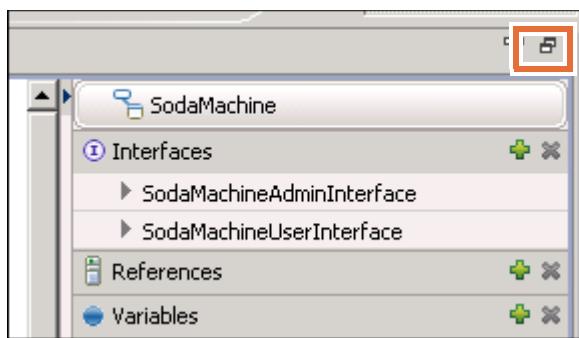
You review many of these artifacts in this exercise before you test the SodaMachine state machine.

All business state machines have one **initial state**. A state represents a planned situation within a business state machine. The business state machine stops execution when it reaches one of two states:

- Final states signal a normal end to a state machine. At least one final state must exist in a business state machine.
- Terminate states signal an abnormal end to a state machine. Terminate states are optional.

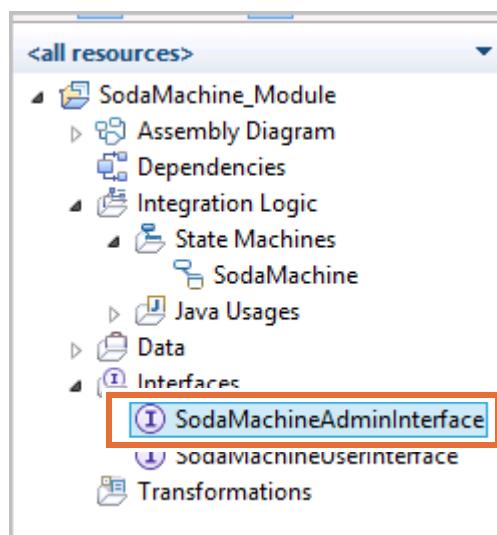
Unlike activities within a parallel activities (flow) structure, a business state machine can be in only one state at any time.

- \_\_\_ d. Click **Restore** at the upper right to return to the original window.



\_\_\_ 3. Explore the **SodaMachineAdminInterface**.

- Just like any business state machine, SodaMachine is a heavily event-driven business process definition. To accept incoming events, the template must provide necessary interfaces with operations to accept incoming messages. Notice that SodaMachine has two interfaces that are associated with it: **SodaMachineAdminInterface** and **SodaMachineUserInterface**. However, remember that a service component in general can have more than one interface; the type of each must be the same (either all Java interfaces, or all WSDL **PortTypes**).
- \_\_\_ a. In the **Business Integration** view, expand **SodaMachine\_Module> Interfaces**, and double-click **SodaMachineAdminInterface**. The interface is shown in the interface editor.



- b. In the **SodaMachineAdminInterface** tab, scroll to the **Operations** area to explore the available operations. Namely, two operations are defined, **PlugIn** and **UnPlug**. The **PlugIn** operation is a request/response that returns a **StateInfo** business object as the output. The **PlugIn** operation is started through Business Process Choreographer Explorer to start an instance of the process.

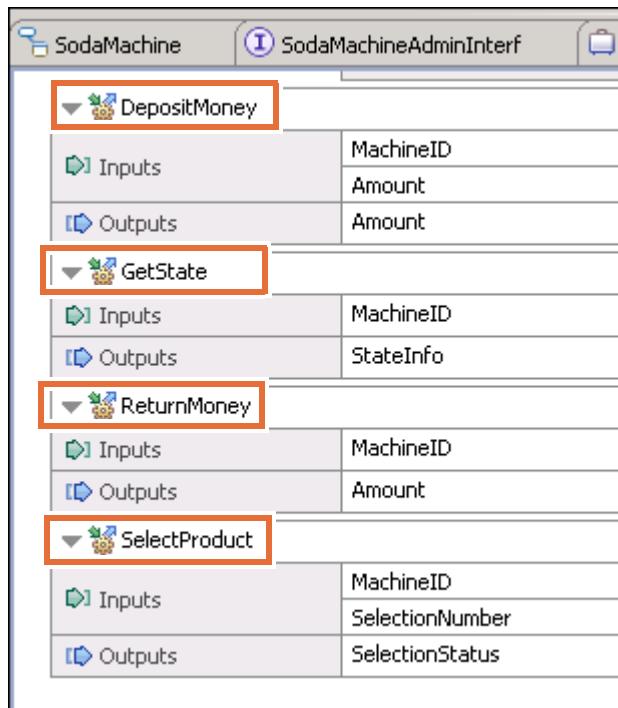
| Name          | Type      |
|---------------|-----------|
| MachineID     | string    |
| Selection1Qty | int       |
| Selection2Qty | int       |
| Selection3Qty | int       |
| Selection4Qty | int       |
| UnitPrice     | double    |
| StateInfo     | StateInfo |

Each soda machine has its own unique ID, along with Selection1Qty, Selection2Qty, Selection3Qty, and Selection4Qty. Each represents the number of available units of a particular soda.

- c. In the Business Integration view, expand **SodaMachine\_Module > Data** and double-click the **StateInfo** business object to open it in the business object editor. The **StateInfo** business object contains the information about **Current state**, along with **MoneyDeposited** and **MoneyReturned**.

|                   |        |
|-------------------|--------|
| e CurrentState    | string |
| e UnitPrice       | double |
| e MoneyDeposited  | double |
| e MoneyReturned   | double |
| e SelectionNumber | int    |
| e Selection1Qty   | int    |
| e Selection2Qty   | int    |
| e Selection3Qty   | int    |
| e Selection4Qty   | int    |

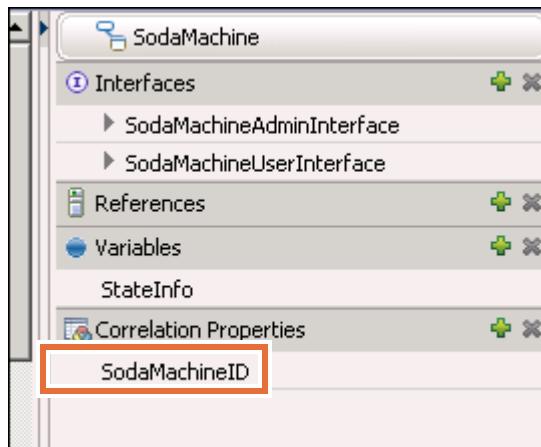
- \_\_\_ d. Feel free to explore other elements of the business object as time allows. Close the interface editor when you are done.
- \_\_\_ 4. Explore the **SodaMachineUserInterface**.
  - \_\_\_ a. In the **Business Integration** view, expand **SodaMachine\_Module > Interfaces**, and double-click **SodaMachineUserInterface**. The interface editor shows an interface.
  - \_\_\_ b. In the **SodaMachineUserInterface** tab, scroll to the **Operations** area to explore the available operations. Four operations are defined.



- \_\_\_ c. Explore the interface elements as time allows. Close the interface editor when you are finished exploring.

\_\_\_ 5. Explore the **SodaMachineID** correlation property.

- Correlation sets are used in runtime environments where multiple instances of the same process are running. The sets allow two partners to initialize a business process transaction, temporarily suspend activity, and then recognize each other again when that transaction resumes.
  - A correlation set for a business process comprises a name and one or more properties. Properties have a name and a type, and can use a property alias to be able to map to specific message parts.
- \_\_\_ a. In the **SodaMachine** state machine editor, click **SodaMachineID** under **Correlation Properties**.

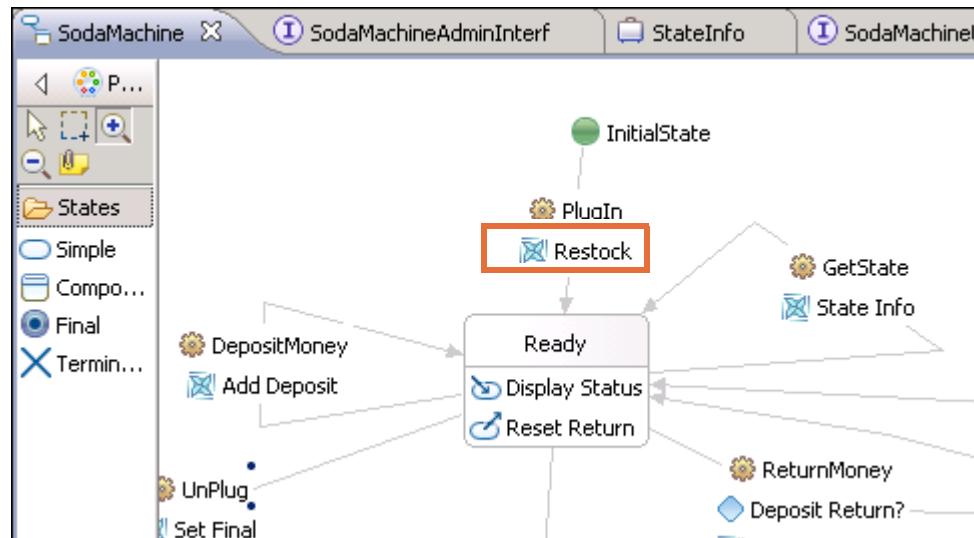


- \_\_\_ b. In the Properties view, explore the details of **SodaMachineID**. Note the XPath expression `/MachineID`, which is defined for the two interfaces.

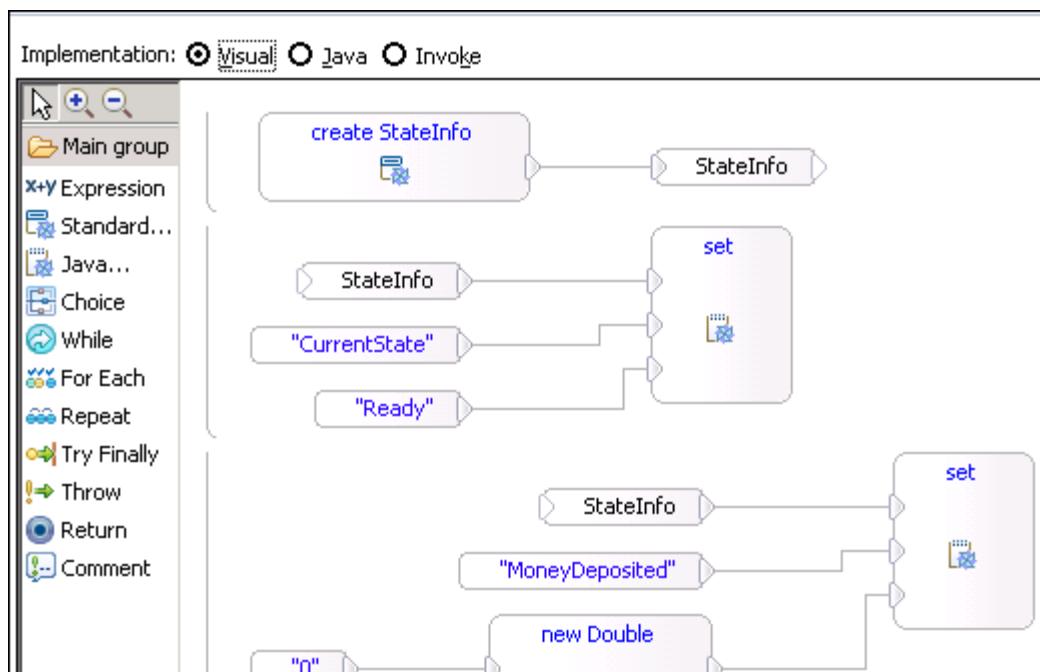
— 6. Explore the **Restock** action.

- The first event into the **SodaMachine** is **PlugIn**, which triggers the creation of a process instance from the SodaMachine template. When the **PlugIn** event is received, the **Restock** action gets run. Recall that an action is an activity that is run when a state is entered, exited, or on a transition within a business state machine. The implementation of this action can be any of the following types:
  - Visual snippet
  - Java code
  - Invoke of an operation that is exposed through a reference

— a. Click the **Restock** action in the editor.



— b. In the **Properties** view, click the **Details** tab to view the visual snippet implementation of the action.



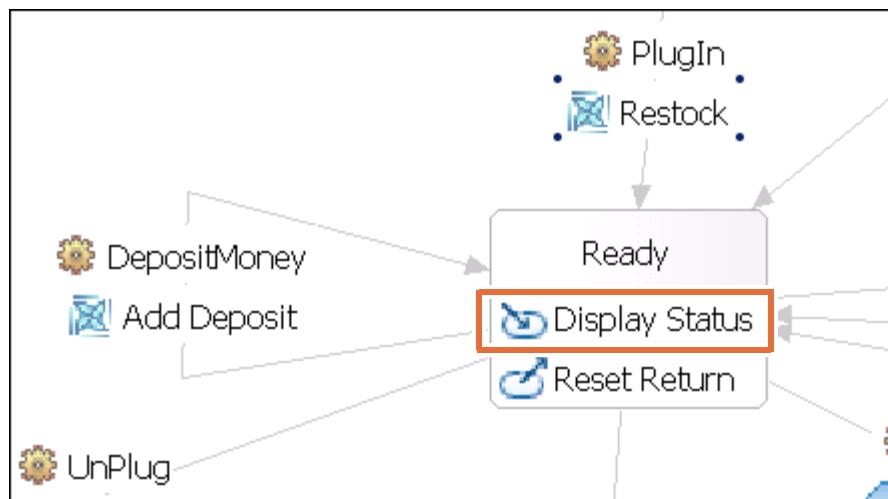
Since the *Restock* action is defined on the transition from the *InitialState* to the *Ready* state, this action gets run before the state changes to *Ready*. The purpose of the *Restock* action is to populate the *StateInfo* business object with its initial values.

— 7. Explore the **Display Status** entry action.

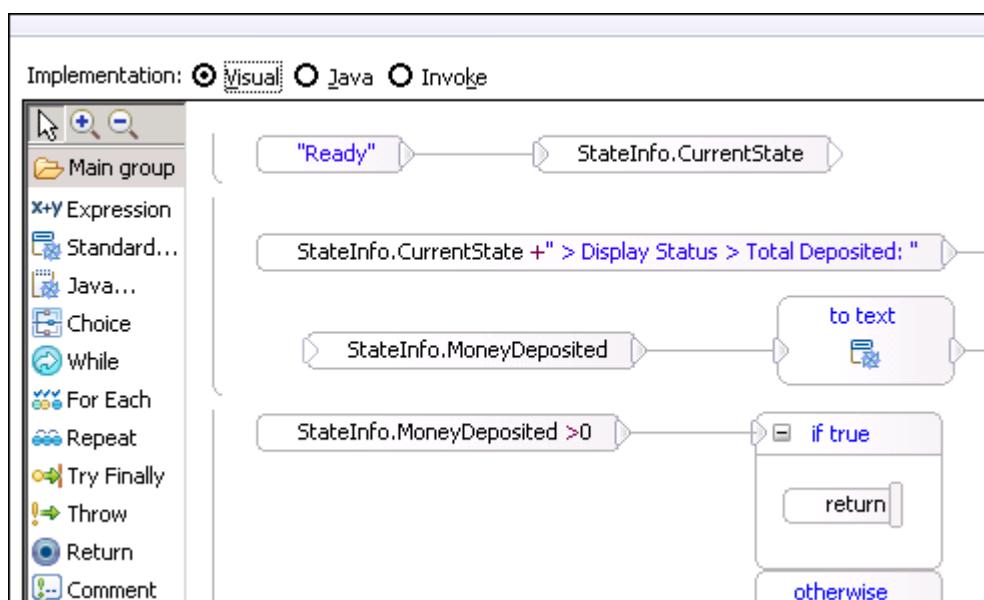
An *entry action* is an activity that the state machine runs when entering in a state. You can add an entry action to intermediate, terminate, or final states. An *exit action* is an activity that the state machine runs when it is leaving a state. You can add an exit action only to intermediate states. Every state can have at most one entry action and one exit action. Entry and exit actions are independent of the transition that is used to enter or exit the state.

In the **SodaMachine** state machine, when the *InitialState* changes to the **Ready** state, the first action that gets run is the **Display Status** entry action.

— a. In the editor, click **Display Status**.

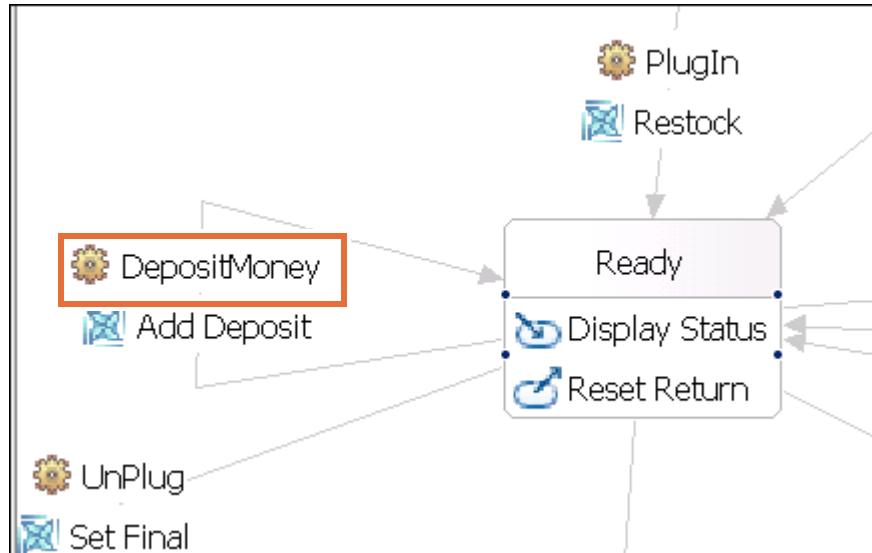


— b. In the **Properties** view, click **Details** to view the visual implementation of the **Display Status** snippet.



The **Ready** state has an entry action that is defined. Therefore, its logic gets run as soon as the process enters the **Ready** state. The logic of the **Display Status** entry action is to print the value of the **StateInfo** variable.

- \_\_\_ 8. Explore the **DepositMoney** event.
  - \_\_\_ a. In the state machine editor, select the **DepositMoney** event.

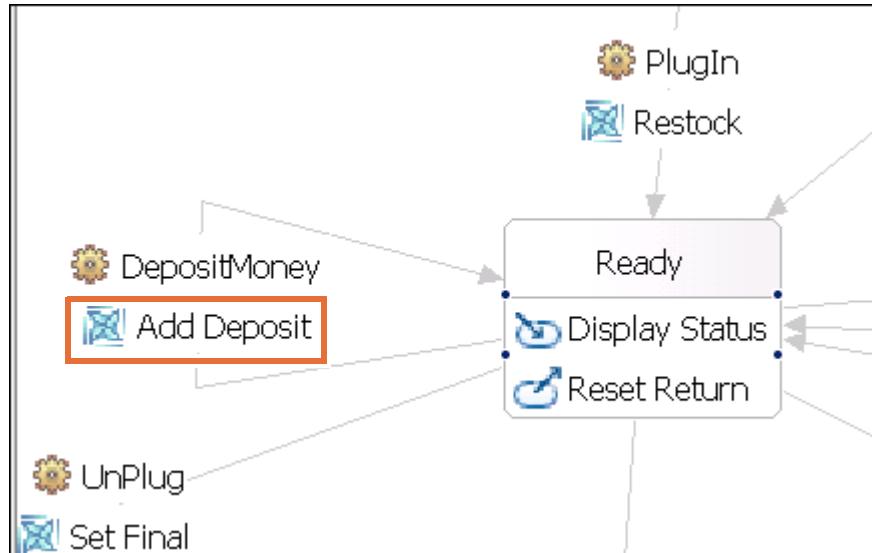


After the soda machine is plugged in and ready for service, a customer must deposit money to purchase a can of soda. The action of depositing money is viewed as an event. The transition channel processes from one state to the next by recognizing the triggering event, evaluating the conditions necessary for control to flow through it, and if processing is allowed, determining what actions can occur. An event can trigger more than one transition at a time, but only one of those transitions fires. The **DepositMoney** event is defined on the transition into the **Ready** state.

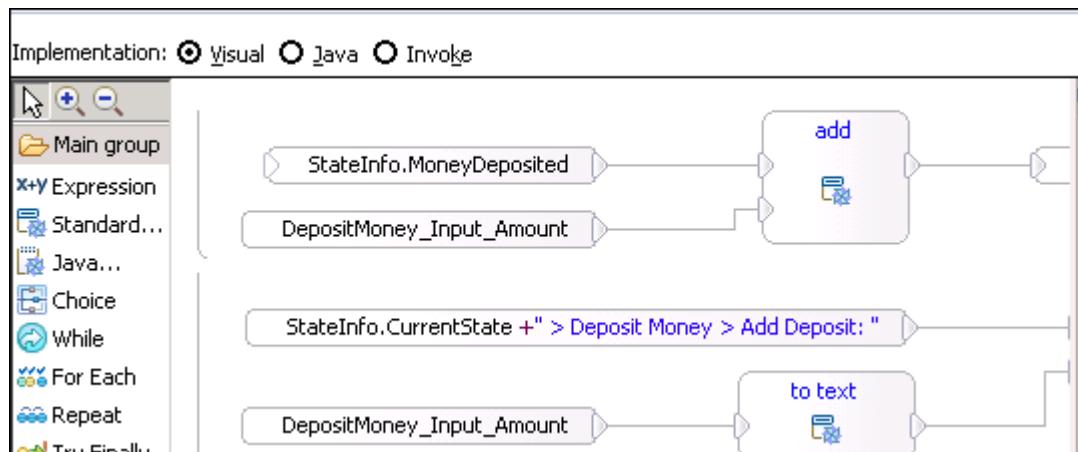
- \_\_\_ 9. Explore the **Add Deposit** action.

Recall that an **action** is an activity that the business state machine runs when it follows a transition. The Add Deposit action follows the **DepositMoney** operation.

- a. In the state machine editor, click the **Add Deposit** action.



- b. In the **Properties** view, click **Details** to view the visual implementation of the **Add Deposit** snippet.

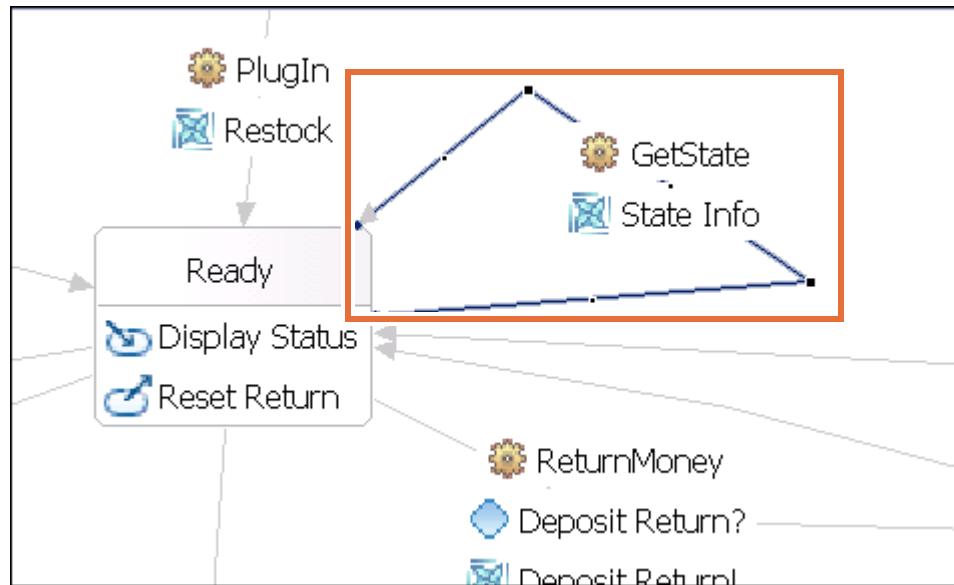


The **Add Deposit** action adds the deposit amount to the **MoneyDeposited** attribute value in the **StateInfo** business object to calculate the new total.

- 10. Explore the **GetState** event.

The **GetState** event is defined for the Ready state.

- a. In the state machine editor, click the transition that contains the **GetState** event. You know that it is selected when the color of the transition gets darker.



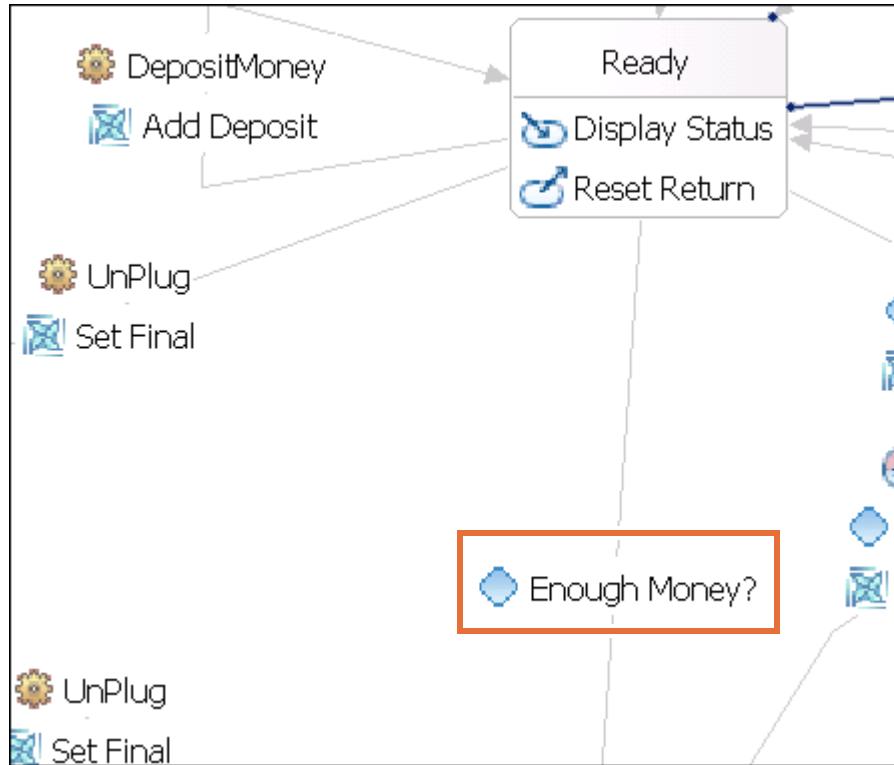
- b. In the **Properties** view, click **Description** to view the Interface and Operation details.

| <b>GetState</b> |                          |
|-----------------|--------------------------|
| Description     | Interface:               |
|                 | SodaMachineUserInterface |
| Operation:      | GetState                 |

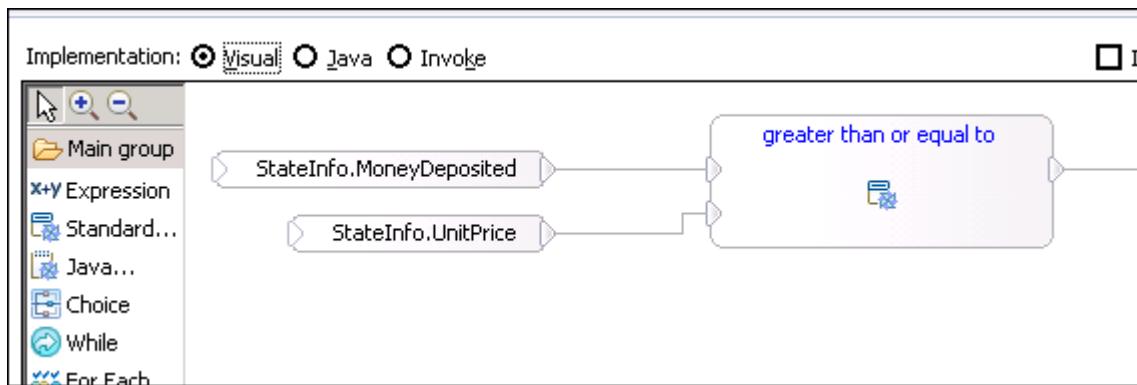
- 11. Explore the **Enough Money** condition.

A **condition** (or *guard*), determines whether a transition must be followed. A condition can be a Boolean expression, a visual snippet, or a call to a reference operation. If the condition evaluates to false, transitions are not followed.

- a. In the state machine editor, click **Enough Money**.



- b. In the **Properties** view, click **Details** to view the visual implementation of the **Enough Money** snippet.



Upon the invocation of the **DepositMoney** operation, the **Add Deposit** action gets run. Set loopback to **Ready**, and then the **Display Status** entry action gets run. The server checks to see whether any automatic transition exists. In this case, an automatic transition goes from the **Ready** state to the **Select** state. An automatic transition is a transition with no operation or action that is defined on it.

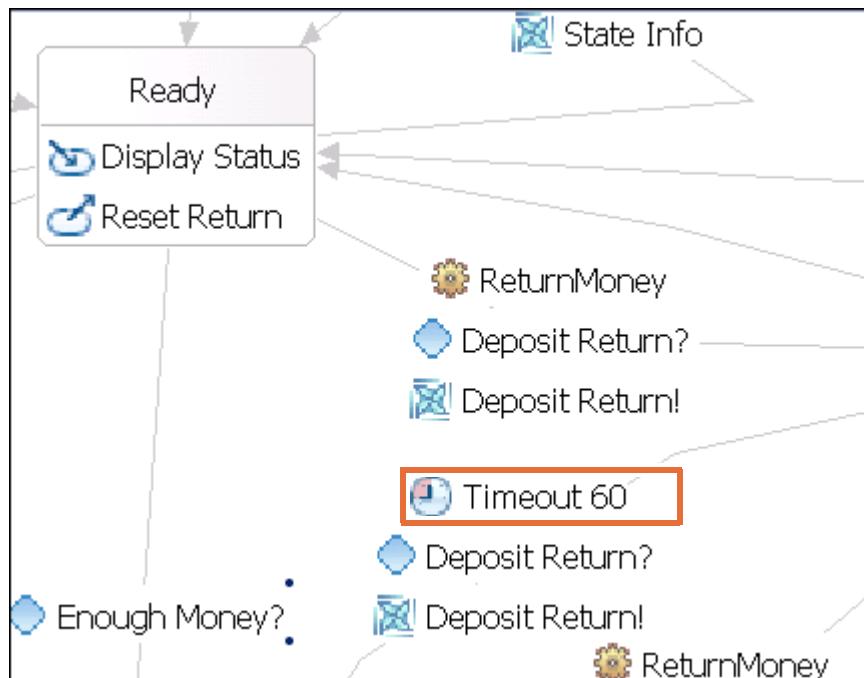
The **Enough Money** condition on the automatic transition gets run only if the condition evaluates to **true**, and then the state moves from **Ready** to **Select**. Therefore, the

**Display Deposit** entry action is defined for the Select state; this entry action gets run when the state is entered.

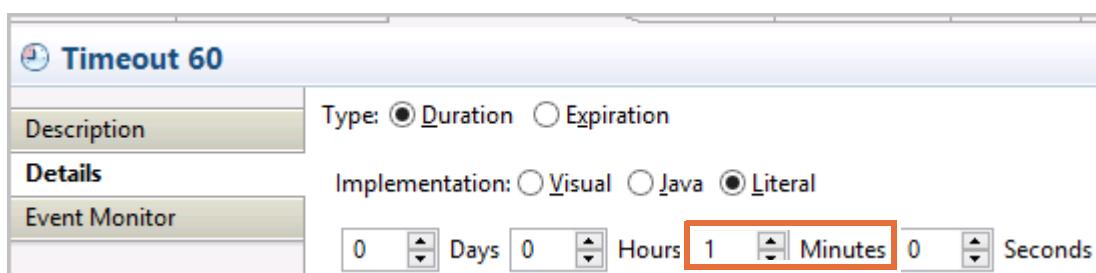
— 12. Explore timeout **Timeout60**.

A timeout imposes a time limit on how long the business state machine must remain in a particular state. It can be an expiration (the date and time when the state expires), or a duration (the length of time to remain in the state). Use a timeout to ensure that the state is not maintained indefinitely while it is waiting for an operation that might never occur.

— a. In the state machine editor, click **Timeout60**.



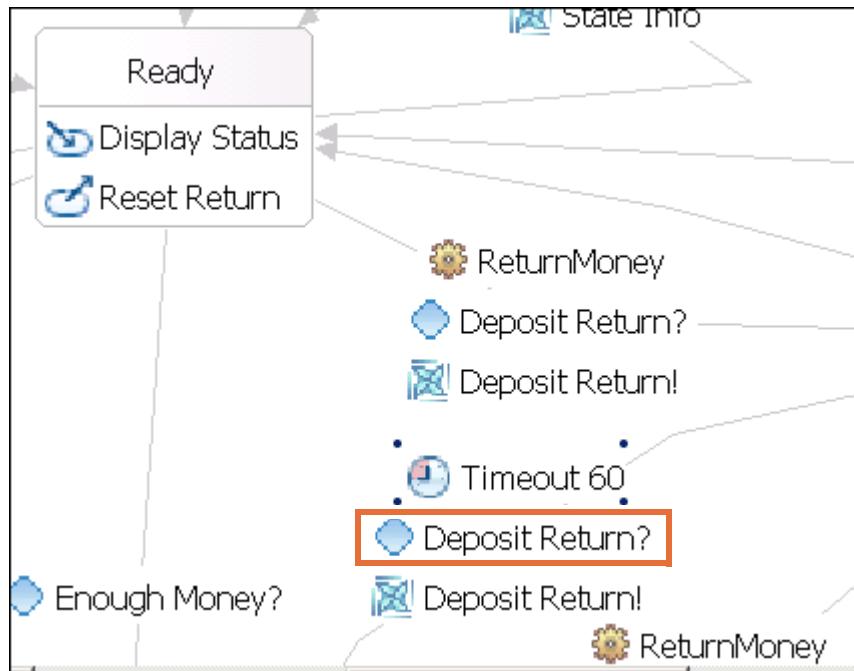
— b. In the **Properties** view, click **Details** to view the implementation of **Timeout 60**.



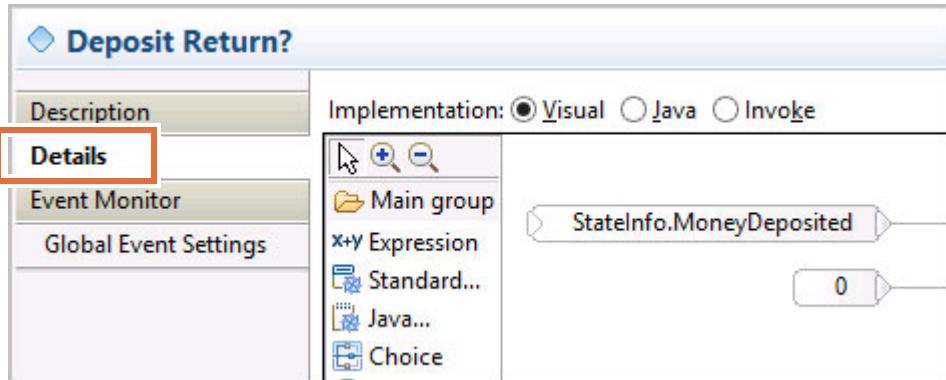
**Timeout** is set to **1 minute**. A time-based event such as a timeout or expiration can also trigger a particular transition out of a state.

— 13. Explore the **Deposit Return** guard.

— a. In the state machine editor, click **Deposit Return**.



— b. In the **Properties** view, click **Details** to view the visual implementation of the **Deposit Return** snippet.



When the timeout occurs, the **Deposit Return** guard is run to check the **MoneyDeposited** amount of the **StateInfo** business object. If the condition evaluates to true, then the **Deposit Return** action is run.

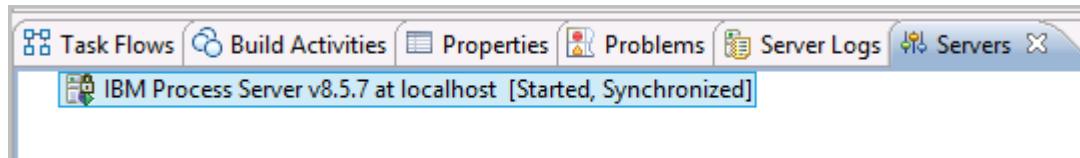
After the deposited money is refunded, the soda machine goes back to its original ready state.

You explored some key pieces of this business state machine. As time allows, feel free to explore other artifacts in the SodaMachine state machine. Be sure to examine the properties of each object.

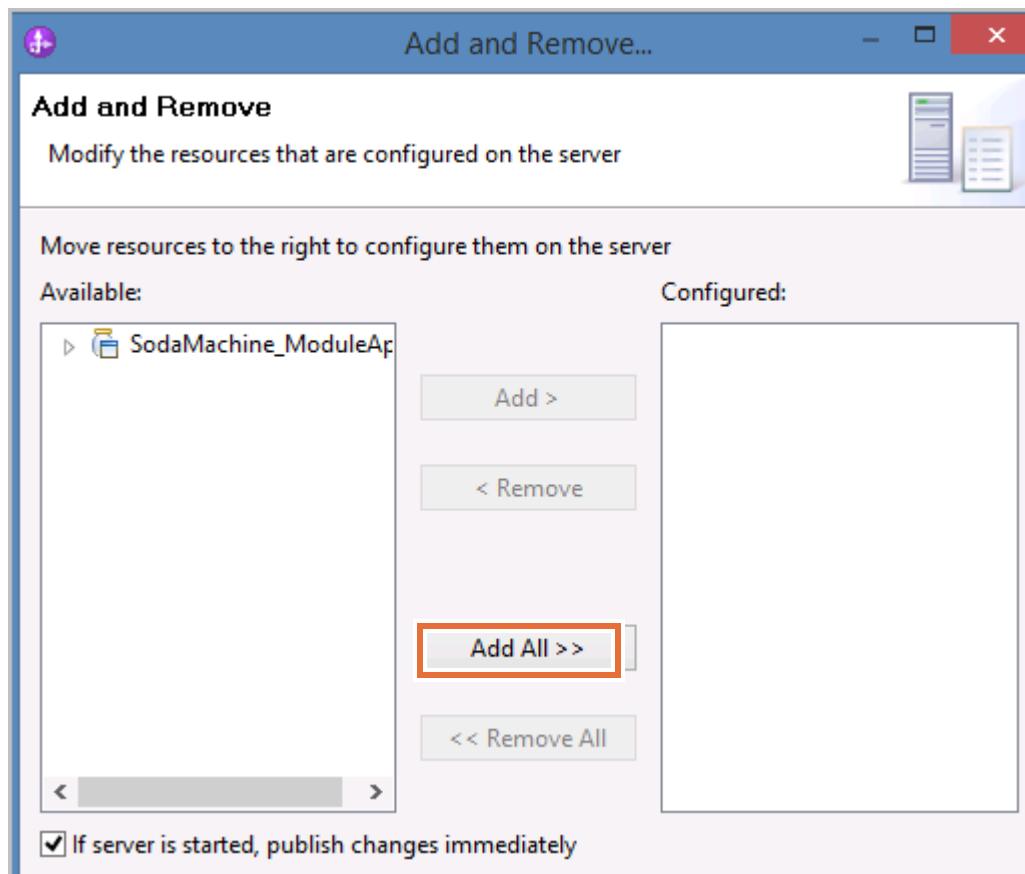
## Part 2: Test the SodaMachine state machine with BPEL Process Choreographer Explorer

You now run some sample scenarios to examine the behavior of the SodaMachine state machine.

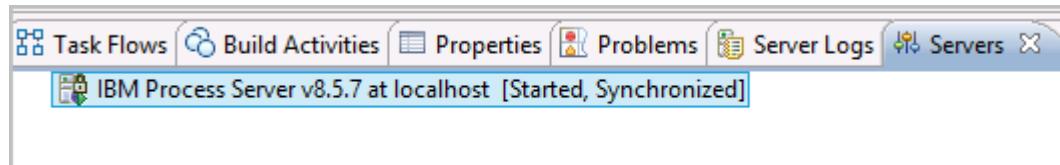
- 1. Start IBM Process Server 8.5.7.
  - a. If your IBM Process Server is not running, then start the server by clicking the **Start the server** icon in the **Servers** view. It might take several minutes for the server to start, depending upon the system resources. After the server starts, the **State** changes to **Started** in the **Servers** view.



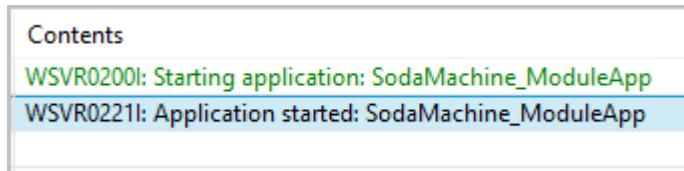
- 2. Install the SodaMachine application on IBM Process Server.
  - a. In the **Servers** view, right-click **IBM Process Server v8.5.7 at localhost** and select **Add and Remove** from the menu.
  - b. Click **Add All** to move **SodaMachine\_ModuleApp** from **Available** to **Configured**.



- \_\_\_ c. Click **Finish** to start the installation. The server **Status** changes to **publishing** while the application is being deployed. After the installation is complete, the server status changes to **[Started, Synchronized]**.



- \_\_\_ d. Click the **Server Logs** tab and confirm that the message that is shown is **Application started: SodaMachine\_ModuleApp**.

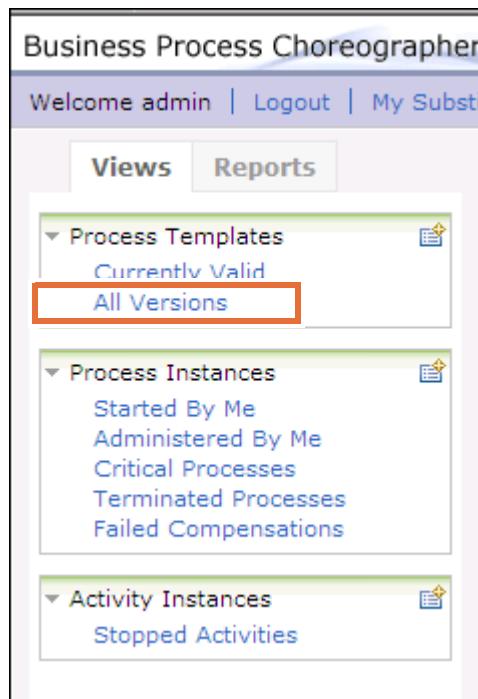


- \_\_\_ 3. Start Business Process Choreographer Explorer.

  - \_\_\_ a. Start an instance of Firefox browser and enter `https://localhost:9443/bpc` in the URL. (You can also right-click the server and choose **Launch > BPC Process Choreographer Explorer**.)
  - \_\_\_ b. At the Welcome page, use `admin` for **User Name** and `web1sphere` for **Password**.
  - \_\_\_ c. Click **Login**.

- \_\_\_ 4. Create a process instance.

  - \_\_\_ a. Under **Process Templates**, click **All Versions**.



- \_\_ b. Click the **SodaMachine** link under **Process Template Name**.

The screenshot shows a search interface for process templates. At the top, there are buttons for 'Instances', 'View Structure', and 'Refresh'. Below these are two search fields: 'Process Template Name' and 'Valid From'. The 'Process Template Name' field contains 'SodaMachine', which is highlighted with a red box. To the right of the search fields is the date '8/4/2014 3:00:00 AM EDT'. At the bottom of the search area, it says 'Items found: 1 Items selected: 0'.

- \_\_ c. Click **Start Instance** to create an instance of the **SodaMachine** process.

The screenshot shows the details of the SodaMachine process template. At the top, there are buttons for 'Start Instance', 'Instances', 'Versions', and 'View Structure'. The 'Start Instance' button is highlighted with a red box. Below these buttons is a section titled 'Process Template Description'. Underneath, it lists the 'Process Template Name' as 'SodaMachine', 'Description' (which is empty), and 'Documentation' (which is also empty).

- \_\_ d. Enter process input data as follows:

- **Process Name:** SodaMachineTest1
- **MachineId:** 1
- **Selection1QTY:** 0
- **Selection2QTY:** 1
- **Selection3QTY:** 5
- **Selection4QTY:** 10
- **UnitPrice:** 0.5

The screenshot shows a form view for the SodaMachineTest1 process instance. At the top, there is a text input field containing 'SodaMachineTest1', which is highlighted with a red box. Below this is a table with six rows, each representing a process input parameter and its value:

|               |     |
|---------------|-----|
| MachineID     | 1   |
| Selection1Qty | 0   |
| Selection2Qty | 1   |
| Selection3Qty | 5   |
| Selection4Qty | 10  |
| UnitPrice     | 0.5 |

The **PlugIn** operation is being invoked to start the instance. It is the same operation in the **SodaMachine** state machine, which you reviewed earlier.

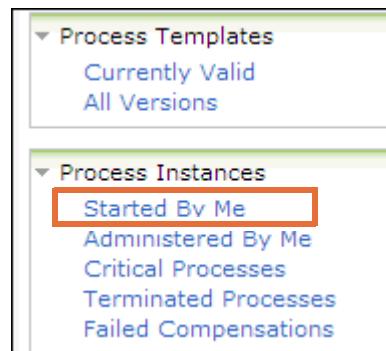
The first event that arrives in the **SodaMachine** is through the **PlugIn** operation, which is started on the **SodaMachineAdminInterface**.

Each soda machine has its own unique ID and SelectionQty, which represents the number of units of a particular soda available.

- \_\_\_ e. Click **Submit** to create the instance.
- \_\_\_ f. Switch to the **Server logs** view in IBM Integration Designer to verify the values that were entered earlier and the **deposit amount** as 0.0.

|                |                                                        |
|----------------|--------------------------------------------------------|
| Aug 8, 2014... | WSVR0200I: Starting application: SodaMachine_ModuleApp |
| Aug 8, 2014... | WSVR0221I: Application started: SodaMachine_ModuleApp  |
| Aug 8, 2014... | Ready > Display Status > Total Deposited: 0.0          |
| Aug 8, 2014... | Ready > Display Status > Selection1Qty: 0              |
| Aug 8, 2014... | Ready > Display Status > Selection2Qty: 1              |
| Aug 8, 2014... | Ready > Display Status > Selection3Qty: 5              |
| Aug 8, 2014... | Ready > Display Status > Selection4Qty: 10             |
| Aug 8, 2014... | Ready > Display Status > UnitPrice: 0.5                |

- \_\_\_ g. In the **Process Instances** section, click **Started by Me**. The newly created SodaMachineTest1 instance is listed.



- \_\_ h. Click the **SodaMachineTest1** link and select the **Process Input Message** tab to view and verify what was entered.

| MachineID     | 1   |
|---------------|-----|
| Selection1Qty | 0   |
| Selection2Qty | 1   |
| Selection3Qty | 5   |
| Selection4Qty | 10  |
| UnitPrice     | 0.5 |

- \_\_ i. Click the **Waiting Operations** tab, and then click **DepositMoney** to send a new event to the business state machine.

| Operation Name      | Port Type Name            | Description | Event Handler |
|---------------------|---------------------------|-------------|---------------|
| PlugIn              | SodaMachineAdminInterface | false       |               |
| GetState            | SodaMachineUserInterface  | false       |               |
| <b>DepositMoney</b> | SodaMachineUserInterface  | false       |               |
| UnPlug              | SodaMachineAdminInterface | false       |               |
| SelectProduct       | SodaMachineUserInterface  | false       |               |

The new input form is shown. Recall that after the soda machine is plugged in and ready for service, a customer must deposit money to purchase a can of soda. Depositing money is an event.

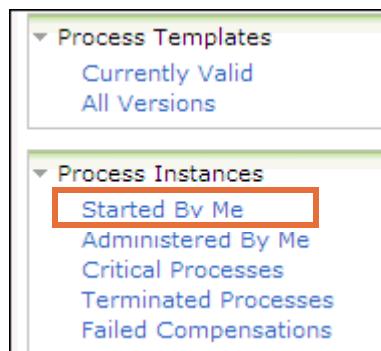
- \_\_ j. Leave the **MachineID** as 1, which is the default **MachineID** value used to instantiate the **SodaMachineTest1**.
- \_\_ k. Enter 0.25 in the **Amount** field.

| MachineID | 1    |
|-----------|------|
| Amount    | 0.25 |

- \_\_ l. Click **Submit**.
- \_\_ m. Switch to the **Server logs** view in IBM Integration Designer to verify that the deposit amount is 0.25.

|                |                                                |
|----------------|------------------------------------------------|
| Aug 8, 2014... | Ready > Display Status > Total Deposited: 0.0  |
| Aug 8, 2014... | Ready > Display Status > Selection1Qty: 0      |
| Aug 8, 2014... | Ready > Display Status > Selection2Qty: 1      |
| Aug 8, 2014... | Ready > Display Status > Selection3Qty: 5      |
| Aug 8, 2014... | Ready > Display Status > Selection4Qty: 10     |
| Aug 8, 2014... | Ready > Display Status > UnitPrice: 0.5        |
| Aug 8, 2014... | Ready > Deposit Money > Add Deposit: 0.25      |
| Aug 8, 2014... | Ready > Display Status > Total Deposited: 0.25 |

- \_\_ n. Switch to the Business Process Choreographer Explorer. In the **Process Instances** section, click **Started by Me**. The SodaMachineTest1 instance is listed.



- \_\_ o. Click the **SodaMachineTest1** link and select the **Process Input Message** tab.
- \_\_ p. Click the **Waiting Operations** tab and then click **DepositMoney** to send a new event to the business state machine.

| Details             | Process Input Message     | Process Output Message | Activities    | Waiting Operations |
|---------------------|---------------------------|------------------------|---------------|--------------------|
| Operation Name      | Port Type Name            | Description            | Event Handler |                    |
| PlugIn              | SodaMachineAdminInterface |                        | false         |                    |
| GetState            | SodaMachineUserInterface  |                        | false         |                    |
| <b>DepositMoney</b> | SodaMachineUserInterface  |                        | false         |                    |
| UnPlug              | SodaMachineAdminInterface |                        | false         |                    |
| SelectProduct       | SodaMachineUserInterface  |                        | false         |                    |

- \_\_ q. Leave the **MachineID** as 1, which is the default **MachineID** value that is used to instantiate the **SodaMachineTest1**.

- \_\_\_ r. Enter 0.5 in the **Amount** field.

The screenshot shows a 'Send message' dialog box. At the top, there are 'Submit' and 'Cancel' buttons. Below them is a section titled 'Activity Input' with a 'Message' sub-section. On the right, there is a 'Form View' window containing two rows of data. The first row has 'MachineID' in the first column and '1' in the second. The second row has 'Amount' in the first column and '0.5' in the second. Both the 'Amount' row and its entire row in the table are highlighted with a red border.

- \_\_\_ s. Click **Submit**.
- \_\_\_ t. Switch to the **Server Logs** view in IBM Integration Designer to verify that the deposit amount is 0.5. Verify that the message in the console indicates the correct amount of deposit. The unit price that is set during the business state instantiation is \$0.50, so adding another \$0.50 to the already deposited \$0.25 provides sufficient funds (\$0.75) to purchase a product.

|                |                                                  |
|----------------|--------------------------------------------------|
| Aug 8, 2014... | Ready > Deposit Money > Add Deposit: 0.25        |
| Aug 8, 2014... | Ready > Display Status > Total Deposited: 0.25   |
| Aug 8, 2014... | Ready > Deposit Money > Add Deposit: 0.5         |
| Aug 8, 2014... | Ready > Display Status > Total Deposited: 0.75   |
| Aug 8, 2014... | Select > Display Deposit > Total Deposited: 0.75 |

- \_\_\_ u. Wait for the **Timeout** to occur, and review the messages in the **Server Logs** view.

|                |                                                  |
|----------------|--------------------------------------------------|
| Aug 8, 2014... | Ready > Display Status > Total Deposited: 0.75   |
| Aug 8, 2014... | Select > Display Deposit > Total Deposited: 0.75 |
| Aug 8, 2014... | Select > Timeout Return Money: 0.75              |
| Aug 8, 2014... | Return > Dispense Cash: 0.75                     |
| Aug 8, 2014... | Ready > Display Status > Total Deposited: 0.0    |
| Aug 8, 2014... | Ready > Display Status > Selection1Qty: 0        |
| Aug 8, 2014... | Ready > Display Status > Selection2Qty: 1        |
| Aug 8, 2014... | Ready > Display Status > Selection3Qty: 5        |
| Aug 8, 2014... | Ready > Display Status > Selection4Qty: 10       |
| Aug 8, 2014... | Ready > Display Status > UnitPrice: 0.5          |

When the timeout of 1 minute occurs, the **Deposit Return** condition is run to check the **MoneyDeposited** amount of the StateInfo business object. Since the condition evaluates to **true**, the **Deposit Return** action is run.

- \_\_\_ 5. Run another test and select a product that is not available for purchase.

This time, you are going to deposit enough money and make the product selection for a product that is unavailable and review the results.

- \_\_\_ a. In the Business Process Choreographer Explorer, click the **SodaMachineTest1** link.
- \_\_\_ b. Click the **Waiting Operations** tab and then click **DepositMoney** to send a new event to the business state machine.
- \_\_\_ c. Leave the **MachineID** as **1**, which is the default **MachineID** value used to instantiate the **SodaMachineTest1**.
- \_\_\_ d. Enter **1.0** in the **Amount** field.

| Form View |     |
|-----------|-----|
| MachineID | 1   |
| Amount    | 1.0 |

- \_\_\_ e. Click **Submit**.
- \_\_\_ f. Observe the messages that the **Entry Actions** option generates on the **Ready** and **Select** states in the **Server Logs** view. The unit price that is set during the business state machine instantiation is \$0.50, so depositing \$1.00 provides sufficient funds to purchase a product.
- \_\_\_ g. In the Business Process Choreographer Explorer, click the **SodaMachineTest1** link.
- \_\_\_ h. Click the **Waiting Operations** tab and then click **SelectProduct** to purchase a can of soda.

- \_\_\_ i. Leave **MachineID** as 1 and enter 1 in the **SelectionNumber** field.

|                 |   |
|-----------------|---|
| MachineID       | 1 |
| SelectionNumber | 1 |

- \_\_\_ j. Click **Submit**.  
 \_\_\_ k. Observe the messages in the **Server Logs** view. You see the message: Product: Unavailable > SelectionNumber: 1.

Upon the arrival of the SelectProduct event, the **Product Unavailable** condition checking (guard) is run. Since it evaluates to **true**, the **Product Unavailable** action is run. Since you initialized the **SodaMachineTest1** with **Selection1Qty** to 0, this condition evaluates to **true**.



### Note

If your session times out, then you can try a deposit and purchase again to see the result. Remember, the timeout is set to 1 minute so if you take more than a minute from the time you made the deposit to the time you make the purchase, it might be too late.

- \_\_\_ 6. Select a product *that is available* for purchase.

This time, you are going to deposit enough money. Select a product that is *available* and review the result.

- \_\_\_ a. In the Business Process Choreographer Explorer, click the **SodaMachineTest1** link.
- \_\_\_ b. Click the **Waiting Operations** tab and then click **DepositMoney** to send a new event to the business state machine.
- \_\_\_ c. Leave the **MachineID** as 1, which is the default **MachineID** value used to instantiate **SodaMachineTest1**.

- \_\_ d. Enter 1.0 in the **Amount** field.

**Send message**

Use this page to send a message to a waiting activity

**Activity Input Message**

| Form View |     |
|-----------|-----|
| MachineID | 1   |
| Amount    | 1.0 |

**Edit Source**

- \_\_ e. Click **Submit**.
- \_\_ f. Observe the messages that the **Entry Actions** generated on the **Ready** and **Select** states in the Server Logs view. The unit price that is set during the business state machine instantiation is \$0.50, so depositing \$1.00 provides sufficient funds to purchase a product.
- \_\_ g. In the Business Process Choreographer Explorer, click the **SodaMachineTest1** link.
- \_\_ h. Click the **Waiting Operations** tab and then click **SelectProduct** to purchase a can of soda.
- \_\_ i. Leave **MachineID** as 1 and enter 2 in the **SelectionNumber** field.

**Activity Input Message**

| Form View       |   |
|-----------------|---|
| MachineID       | 1 |
| SelectionNumber | 2 |

**Edit Source**

- \_\_ j. Click **Submit**.

- \_\_ k. Observe the messages that are generated in the **Server Logs** view.

|                |                                                    |
|----------------|----------------------------------------------------|
| Aug 8, 2014... | Ready > Deposit Money > Add Deposit: 1.0           |
| Aug 8, 2014... | Ready > Display Status > Total Deposited: 1.0      |
| Aug 8, 2014... | Select > Display Deposit > Total Deposited: 1.0    |
| Aug 8, 2014... | Select > Product Available > SelectionNumber: 2    |
| Aug 8, 2014... | Dispense > Calculate Change: 0.5                   |
| Aug 8, 2014... | Dispense > Adjust Quantity > Selection Quantity: 0 |
| Aug 8, 2014... | Return > Dispense Cash: 0.5                        |
| Aug 8, 2014... | Ready > Display Status > Total Deposited: 0.0      |
| Aug 8, 2014... | Ready > Display Status > Selection1Qty: 0          |
| Aug 8, 2014... | Ready > Display Status > Selection2Qty: 0          |
| Aug 8, 2014... | Ready > Display Status > Selection3Qty: 5          |

The **SelectProduct** event is defined on the transition link between the **Select** and **Dispense** states as well. If the **Product Unavailable** condition evaluates to **false**, then the **Product Available** condition is run. When you select a soda with a quantity greater than 0, the **Product Available** action is run. That action is what happened in this step.

The **dispense** state has the **Calculate Change** entry action defined. The **MoneyReturn** attribute of the **StateInfo** business object gets assigned the value of **MoneyDeposited** subtracted from **UnitPrice**. After the successful execution of the Calculate Change entry action, the server checks to see whether an automatic transition exists. If the **MoneyReturned** value is greater than 0, the **Change Return** condition returns **true**. If the **MoneyReturned** value is 0, the **No Change** condition returns **true**.

After the condition for **Adjusted Quantity** evaluates to true, the state exits out of the **Dispense** state. Therefore, the exit action gets run. Since a product was successfully purchased, the soda machine has one less of the purchased item. However, you deposited a total amount of \$1.00, and the unit price is \$0.50. Therefore, the **StateInfo.MoneyReturned** value is greater than 0.

Finally, the transition between **Return** and **Ready** is automatic. Therefore, the process instance does not wait for an incoming event, and it moves directly to the **Ready** state.

- \_\_ 7. Run the **UnPlug** operation on the SodaMachine state machine.
- \_\_ a. In the Business Process Choreographer Explorer, click the **SodaMachineTest1** link.

- \_\_ b. Click the **Waiting Operations** tab and then click **UnPlug**.

| Operation Name | Port |
|----------------|------|
| PlugIn         | Soda |
| GetState       | Soda |
| DepositMoney   | Soda |
| <b>UnPlug</b>  | Soda |
| SelectProduct  | Soda |
| ReturnMoney    | Soda |

Items found: 6    <<

On the arrival of an **UnPlug** event, the **Set Final** action is run.

- \_\_ c. Enter 1 in the **MachineID** field to submit a new user event to finalize the SodaMachine business state machine.

Activity Input Message

**Submit**   **Cancel**

Form View

|           |   |
|-----------|---|
| MachineID | 1 |
|-----------|---|

**Edit Source**

- \_\_ d. Click **Submit** and observe the new messages that are generated in the **Server Logs** view. Verify that the **SodaMachineTest1** instance moved to the final state. The business state machine instance stays in the server until it moves to the final state. On the arrival of the **UnPlug** event, this soda machine instance moves to its final state, and the instance is terminated.

|                |                                               |
|----------------|-----------------------------------------------|
| Aug 8, 2014... | Final > Display Status > Total Deposited: 0.0 |
| Aug 8, 2014... | Final > Display Status > Selection1Qty: 0     |
| Aug 8, 2014... | Final > Display Status > Selection2Qty: 0     |
| Aug 8, 2014... | Final > Display Status > Selection3Qty: 5     |
| Aug 8, 2014... | Final > Display Status > Selection4Qty: 10    |
| Aug 8, 2014... | Final > Display Status > UnitPrice: 0.5       |

- \_\_ 8. Click the **Logout** link to log out of the Business Process Choreographer Explorer.  
\_\_ 9. Close the browser window.

- \_\_\_ 10. Remove the applications from the server and stop the server.
  - \_\_\_ a. In the **Servers** view, right-click **IBM Process Server v8.5.7 at localhost** and choose **Add and Remove** from the menu.
  - \_\_\_ b. Click **Remove All** and click **Finish**.
- \_\_\_ 11. Close the IBM Integration Designer. Click **File > Exit**, or click **X** in the upper-right corner.

## End of exercise

# Exercise 7. Defining transactional behavior in SCA applications

## Estimated time

01:00

## Overview

In this exercise, you view the transaction settings in the Account Opening end-to-end lab scenario. You view the transaction propagation settings in the integration solution diagram of the scenario, and you view propagation settings for an individual module with the assembly diagram. After viewing the transaction propagation settings, you examine the transaction boundaries around activities in a BPEL process.

## Objectives

After completing this exercise, you should be able to:

- View transaction propagation settings for a composite application by using an integration solution diagram
- View transaction propagation settings for an individual module by using the assembly diagram
- Examine the transaction boundaries for activities in long-running business processes
- Use a scope in a long-running business process to manage persistence

## Introduction

In this exercise, you view transaction propagation settings for the account opening lab scenario (a composite application) by using the integration solution diagram for the scenario. You also view how transactions are propagated in the `FoundationModule` (the core of the account opening scenario) with the assembly diagram for the module. After you view the transaction propagation settings, you examine the transaction behavior settings on activities in the `AccountVerification` business process. Finally, you compartmentalize activities from the business process into a scope so that you can control the persistence and manage the transaction settings on those activities.

## Requirements

Completing the exercises for this course requires a lab environment that includes the exercise support files, Mozilla Firefox, IBM Integration Designer V8.5.7 Advanced, IBM DB2 V10.1, and the IBM Process Server V8.5.7 test environment.

## Exercise instructions

In this exercise, you examine transaction settings and set transaction-related quality of service qualifiers. Setting up transactions in an SOA environment can be an intricate exercise. The assembly editor, BPEL editor, and integration solution diagram provide aids to make it easier to understand the way transactions are processed. However, you must understand the mechanics of transactions before you try to create them or alter the default settings.

When you establish a transaction, you want to have confidence that all the operations commit successfully or roll back completely. You must ensure that a transaction can propagate to all components that must participate in the unit of work. Quality of service qualifiers and invocation style control the propagation of transactions in a module. A number of conditions must exist for a transaction to propagate successfully:

- A transaction qualifier must be set on the implementation, and the qualifier must specify a global transaction.
- The implementation must be started synchronously.
- The component must join a propagated transaction.
- The transaction cannot be suspended on any of the participating references.
- The preferred interaction style on a participating interface cannot influence the invocation style to become asynchronous.



### Important

Given the nature of this sample environment, it is not feasible to test the theories that are practiced in this exercise. No method is available by which you can easily discern or detect the subtle changes that are established by manipulating transaction boundaries. Therefore, this exercise has no test at the end of it.

### ***Part 1: View transaction propagation settings for a composite application by using an integration solution diagram***

You can view transaction settings in applications that are developed in IBM Integration Designer in several ways. You can view transaction propagation settings in the assembly editor and in integration solution diagrams. In this portion of the exercise, you examine transaction settings in the account verification lab scenario.

To view transaction settings in an integration solution diagram:

- 1. Open the Exercise 7 workspace.
  - a. On your desktop, open the folder that is labeled **Exercise Shortcuts**.
  - b. Double-click the shortcut that is labeled **Exercise 7**.
  - c. Close the **Getting Started** tab.
- 2. Open the `Create_Account_Solution` integration solution diagram and turn on highlighting for transactions.

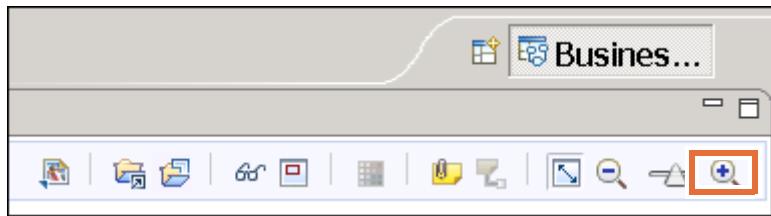
In the integration solution editor, you can show and highlight the transaction path between components in a single module or between the components in all of the modules that make up your integration solution. You can also show and highlight the transaction path from a specific component to the other components with which it interacts.

- a. In the Business Integration view, expand `Create_Account_Solution` and double-click **Solution Diagram**.



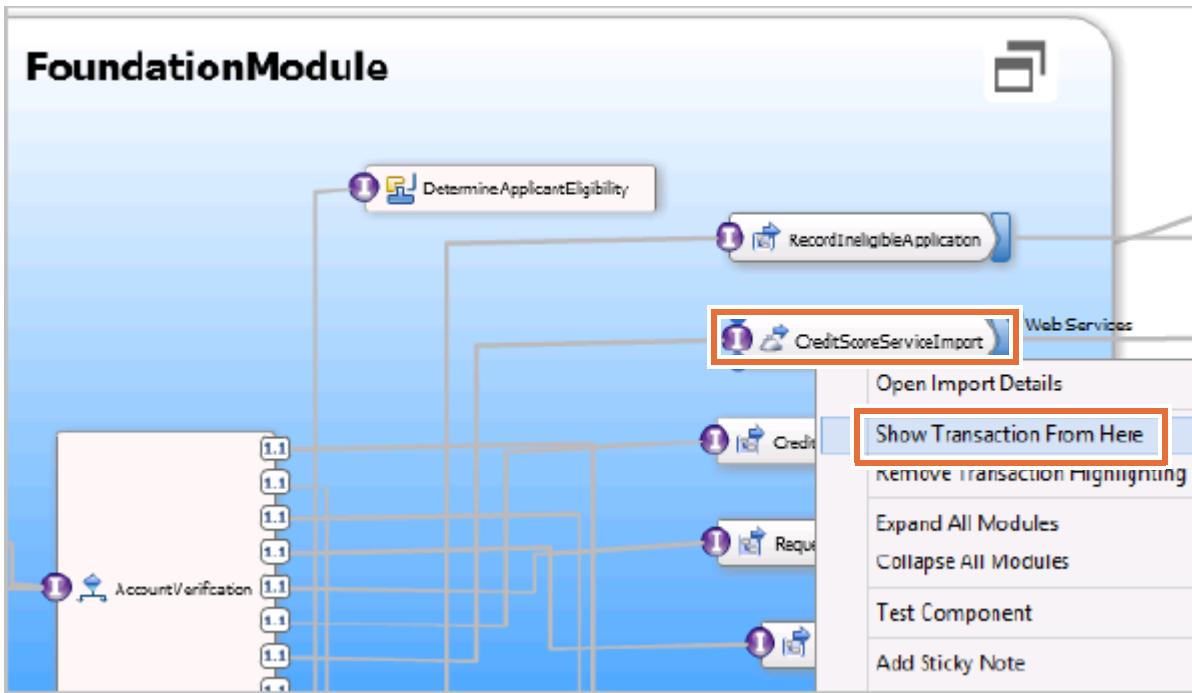
### Note

To increase the viewing area for the diagram, double-click the solution diagram tab to maximize it and use the **Zoom In** icon to enlarge it.

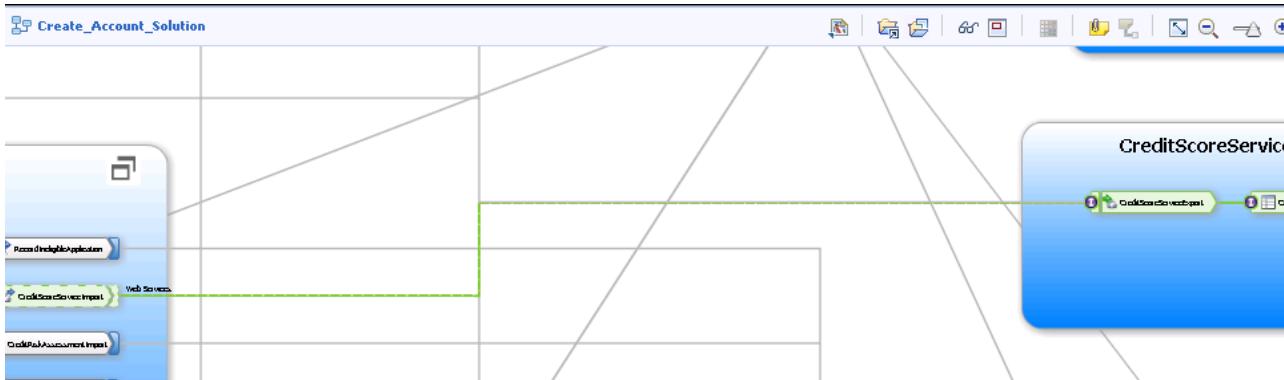


- b. You can right-click any module and choose **Show Transaction In Module** from the menu to see transaction boundaries for a single module. Or you can right-click the component and select **Show Transaction From Here** when you want to show the transaction path between a specific component and the components with which it interacts.
- 3. Examine the highlighting for transaction propagation in the `Create_Account_Solution` integration solution diagram.
- a. In the `Create_Account_Solution` integration solution diagram, right-click anywhere on the canvas and choose **Expand All Modules**.

- \_\_\_ b. Right-click **CreditScoreServiceImport** in **FoundationModule** and choose **Show Transaction From Here** from the menu to see transaction boundaries for a component.

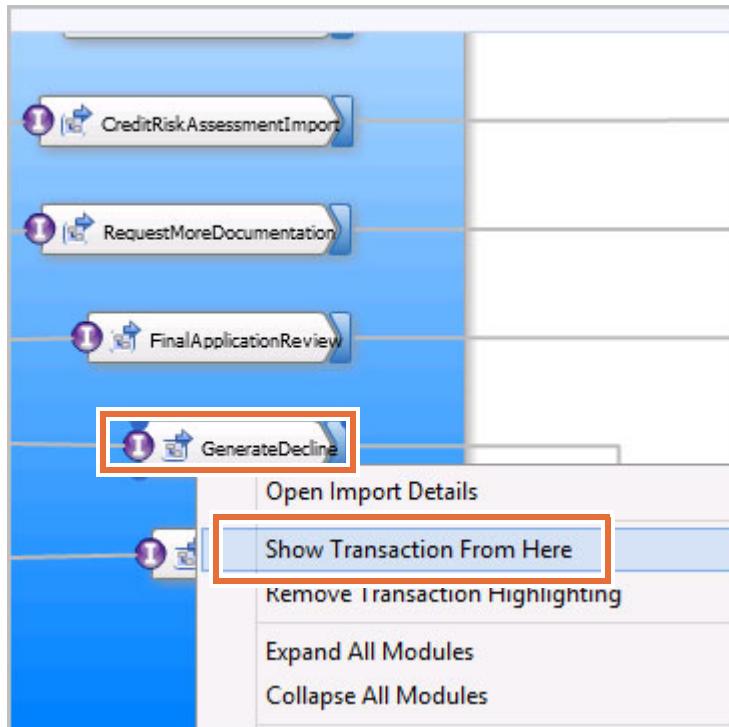


- \_\_\_ c. In **FoundationModule**, view the highlighting for transactions along the connector from the **CreditScoreServiceImport** component to the **CreditScoreService** module. A dashed green line represents the transaction propagation in this case. A dashed green line along a wire indicates that some situations propagate and join a transaction, while some do not.

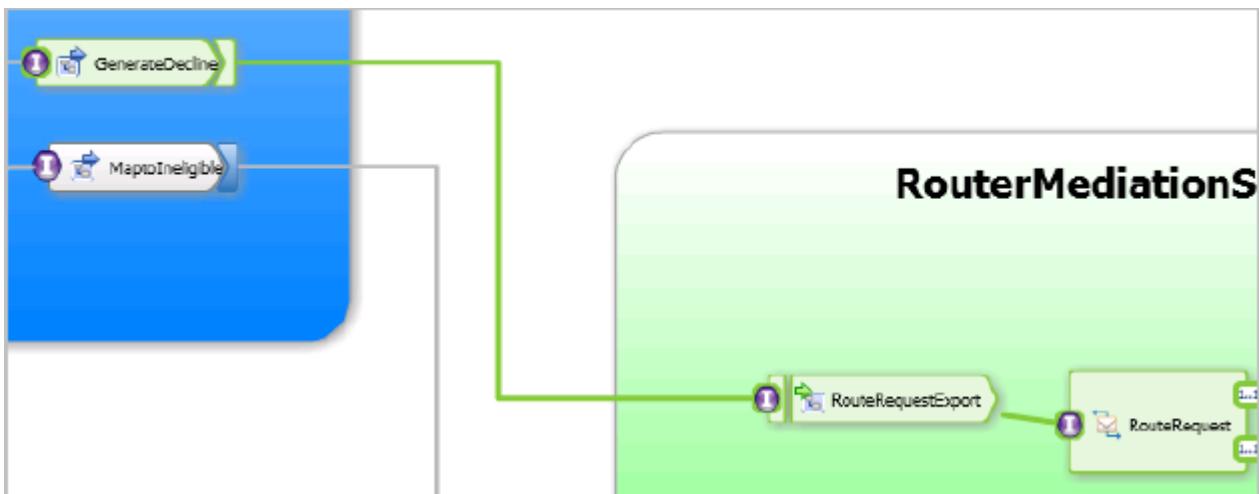


- \_\_\_ d. Right-click anywhere on the canvas and select **Remove Transaction Highlighting**.

- e. In the `Create_Account_Solution` integration solution diagram, right-click **GenerateDecline** in `FoundationModule`, and choose **Show Transaction From Here** from the menu to see the transaction boundaries for a component.



- f. In `FoundationModule`, view the transaction highlight along the connector from the `GenerateDecline` import component to the `RouterMediationService`. A solid green line represents transaction propagation in this case. A solid green line along a wire indicates that the component propagates and joins a transaction.





## Information

In general, a solid green line around an item or along a wire indicates that the existing transaction is propagated or joined. A dotted line around an object or along a wire indicates that a transaction can be joined or propagated. The absence of transaction highlights indicates that no transaction is propagated or joined; the component runs in its own transaction.

For more information about transaction settings for specific components such as interfaces or references, see the IBM Knowledge Center topic: "Defining transactional behavior."

---

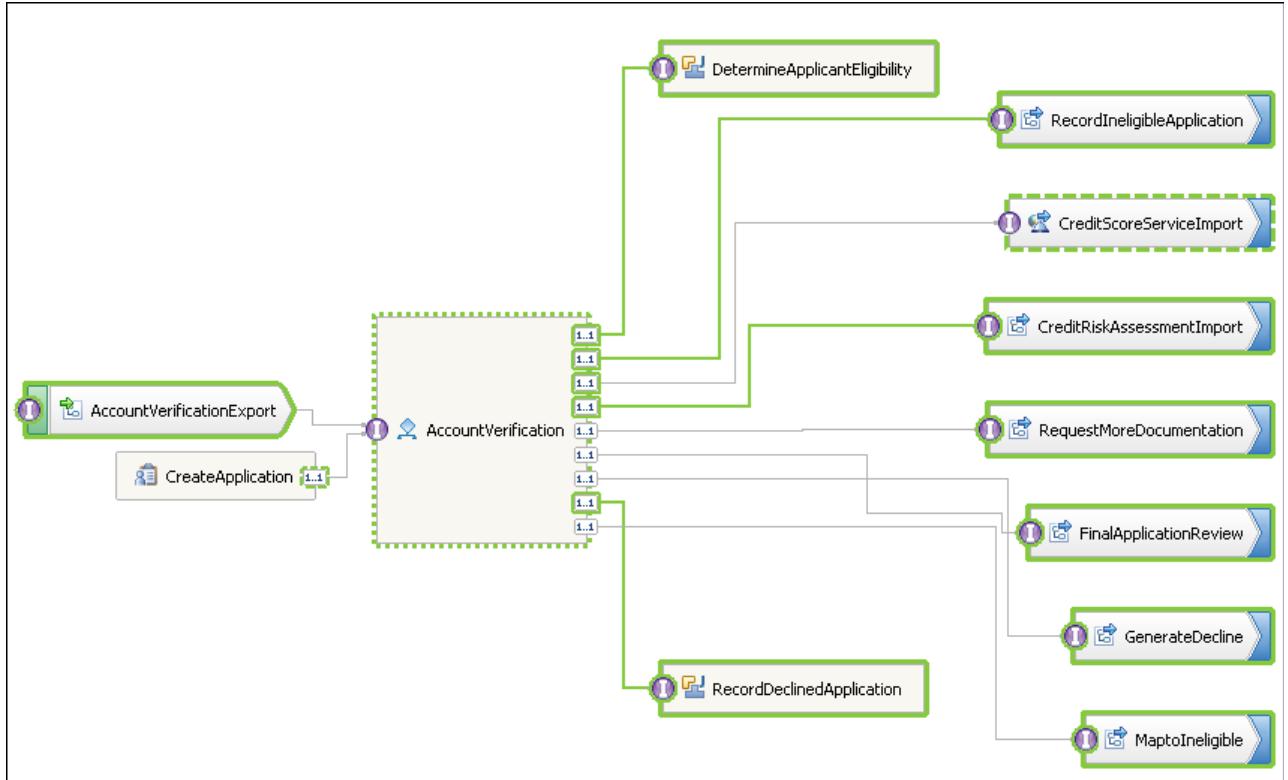
### **Part 2: View transaction propagation settings for an individual module by using the assembly diagram**

#### **Assembly diagram (visual approach)**

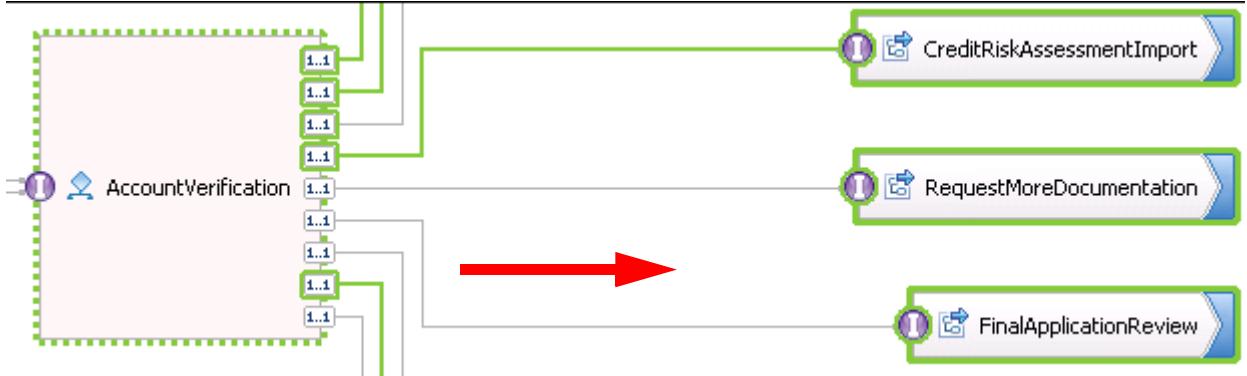
To view transaction settings in an assembly diagram:

- \_\_\_ 1. Open the `FoundationModule` assembly diagram and turn on transaction highlights.
- \_\_\_ a. In the `Create_Account_Solution` integration solution diagram, right-click `FoundationModule` and choose **Open Assembly Diagram** from the menu.

- b. When the assembly diagram loads, right-click the canvas and choose **Show Transaction Highlighting** from the menu. Turning on transaction highlights gives you visual indicators of transaction propagation in the module (showing the highlights is not the same as showing transaction boundaries). Your diagram resembles the following screen capture:

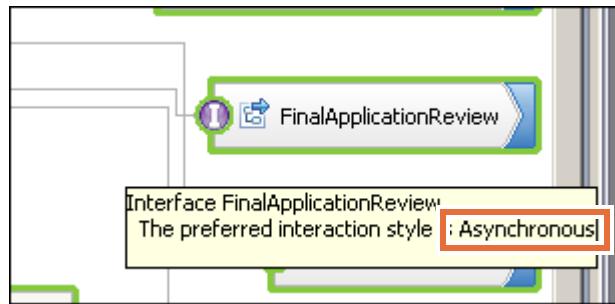


- 2. Examine the transaction settings in the `FoundationModule` assembly diagram.
- a. Examine the transaction settings from the `AccountVerification` BPEL component to the `FinalApplicationReview` import. No transaction highlight is shown along the wire from the `AccountVerification` reference to the `FinalApplicationReview` interface.



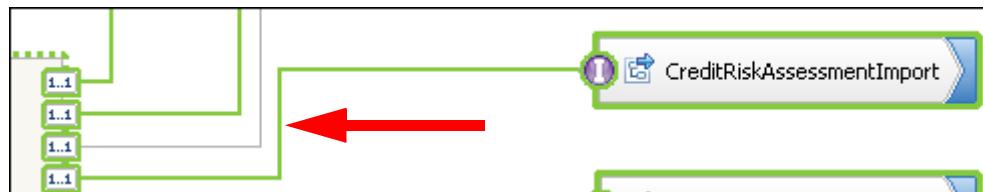
`FinalApplicationReview` is an import component that calls an export in the `HumanTaskServices` module. This export is wired to a stand-alone human task that allows an employee to do a final review of a customer application.

- \_\_\_ b. Hover the mouse pointer over the `FinalApplicationReview` import component. The dialog indicates that the task is started asynchronously. This setting is determined by using the preferred interaction style in the WSDL interface.

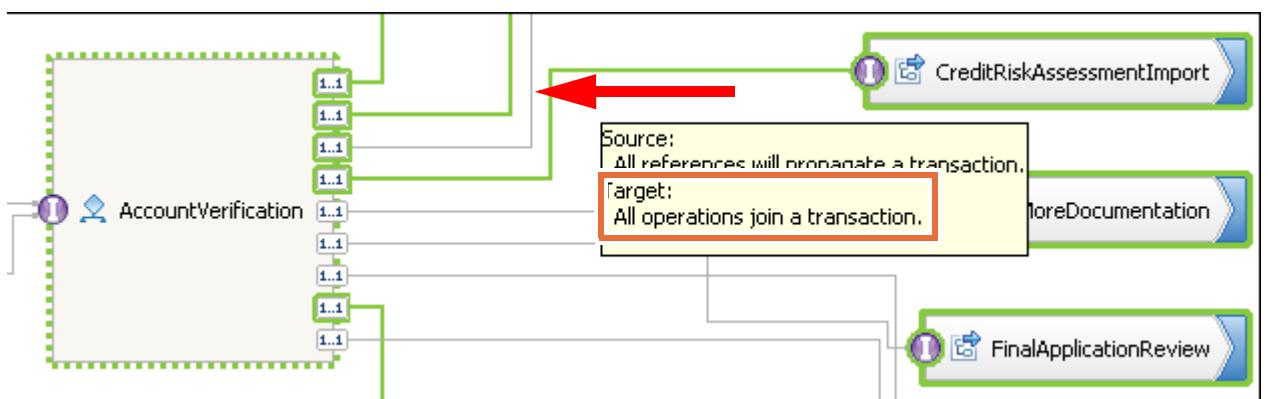


Because `FinalApplicationReview` is a human task, the global transaction that is represented with the `AccountVerification` business process cannot be propagated to it. The absence of highlighting from `AccountVerification` to `FinalApplicationReview` indicates that the human task is started asynchronously or runs in its own transaction. Human tasks in IBM Integration Designer are always started asynchronously.

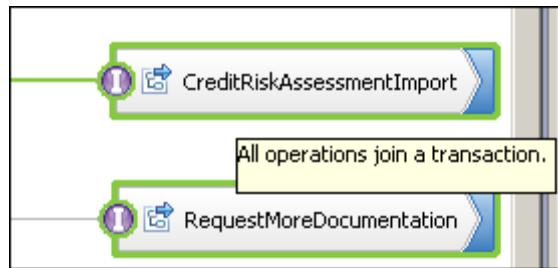
- \_\_\_ c. Examine the transaction settings from `AccountVerification` to the `CreditRiskAssessmentImport` component. The transaction highlight along the wire is a solid green line. The transaction is propagated to the `CreditRiskAssessmentImport`.



- \_\_\_ d. Hover the mouse pointer over the wire that leads from `AccountVerification` to `CreditRiskAssessmentImport`. The dialog indicates that the reference is set to propagate the transaction and that the target operation is set to join this transaction.



- e. Hover the mouse pointer over the `CreditRiskAssessmentImport` component. The dialog indicates that the interface operation started is set to join in the transaction that is propagated from `AccountVerification`.



- f. As time allows, examine the transaction settings and preferred interaction styles for the other components on the `FoundationModule` assembly diagram. You can also examine the transaction settings on one of the mediation module assembly diagrams.

## Assembly diagram (detailed approach)

Viewing transactions in an assembly diagram is useful to graphically determine the flow of transactions between SCA components. However, this visual method does not provide enough details about the transactions and their settings. With the detailed approach, you can view the individual transaction settings between SCA components.

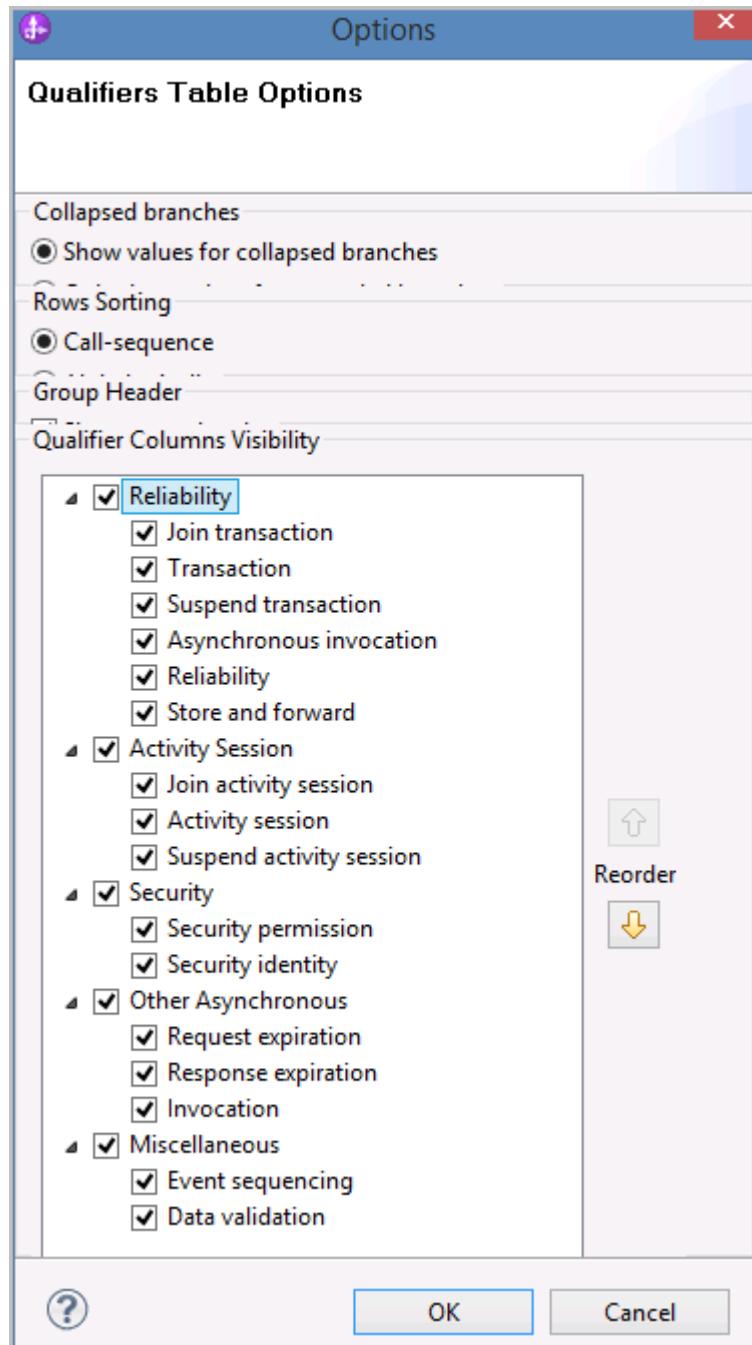
- 1. With the assembly diagram still open, click the **Properties** tab.

- 2. Click the tab that is named **All Qualifiers**. The quality of service (QoS) qualifiers are captured in a table. The SCA components are along the vertical axis. The QoS qualifiers are along the horizontal axis. These qualifiers are divided into groups:
- Reliability
  - Activity session
  - Security
  - Other asynchronous
  - Miscellaneous

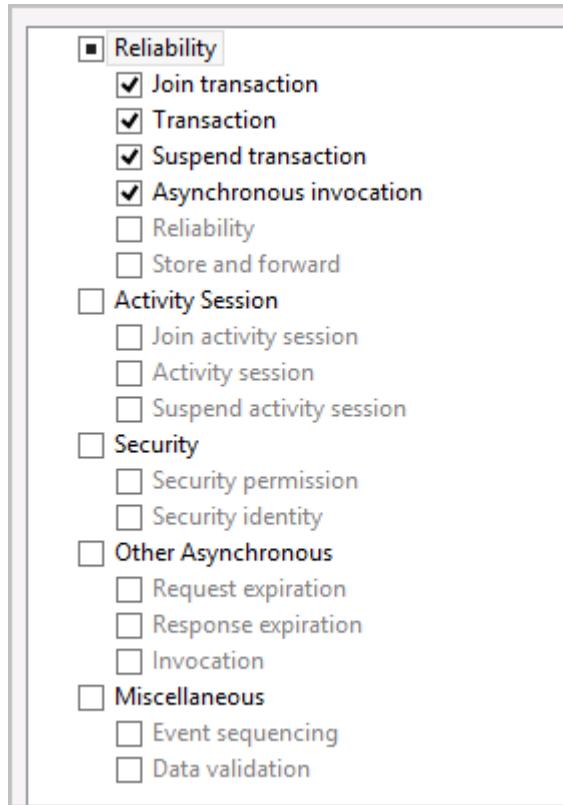
The following table shows the qualifiers that determine the Quality of Service.

| All Qualifiers                | Reliability Qualifiers | Activity Session Qualifiers | Security Qualifiers | Other Asynchronous Qualifiers |
|-------------------------------|------------------------|-----------------------------|---------------------|-------------------------------|
| Component                     | Join t...<br>T...      | Join a...<br>Activi...      | Secur...<br>Requ... | Requ...                       |
| + AccountVerification         |                        |                             |                     |                               |
| + AccountVerification <False> | <False>                | <...                        |                     |                               |
| + DetermineApplicar           | <True>                 | <...                        |                     |                               |
| + RecordIneligibleAp          | <True>                 |                             |                     |                               |
| + CreditScoreService          | <False>                |                             |                     |                               |
| + CreditRiskAssessm           | <True>                 |                             |                     |                               |
| + RequestMoreDocu             | <True>                 |                             |                     |                               |
| + FinalApplicationRe          | <True>                 |                             |                     |                               |
| + GenerateDecline             | <True>                 |                             |                     |                               |
| + RecordDeclinedAp            | <True>                 | <...                        |                     |                               |
| + MaptoIneligible             | <True>                 |                             |                     |                               |
| + CreateApplication           |                        | <...                        |                     |                               |

- 3. Along the top of the table, you see filters that you can use to quickly view the contents of the **All Qualifiers** table. For instance, if you click **Reliability Qualifiers**, only the Reliability QoS qualifiers from the table are shown.
- a. Click **Options**. The **Qualifiers Table Options** dialog opens.



- 4. With the **Qualifiers Table Options** dialog, you can select which qualifiers are shown in the table. By default, all qualifiers and their table headers are selected. Clear all the qualifiers under the **Activity Session**, **Security**, **Other Asynchronous**, and **Miscellaneous** headings. Clear the **Reliability** and **Store and forward** qualifiers under the **Reliability** header.



- 5. Click **OK**. Only those selected qualifiers are shown in the table. You can choose to resize the columns so the names of the qualifiers are visible.

| Component                       | Reliability      |             |                     |                         |
|---------------------------------|------------------|-------------|---------------------|-------------------------|
|                                 | Join transaction | Transaction | Suspend transaction | Asynchronous invocation |
| + AccountVerificationExport     | <False>          | <Global>    | <False>             | <Commit>                |
| + AccountVerification           | <True>           | <Global>    |                     |                         |
| + DetermineApplicantEligibility | <True>           |             |                     |                         |
| + RecordIneligibleApplication   |                  |             |                     |                         |
| + CreditScoreServiceImport      | <False>          |             |                     |                         |
| + CreditRiskAssessmentImport    | <True>           |             |                     |                         |
| + RequestMoreDocumentation      | <True>           |             |                     |                         |
| + FinalApplicationReview        | <True>           |             |                     |                         |
| + GenerateDecline               | <True>           |             |                     |                         |
| + RecordDeclinedApplication     | <True>           | <Global>    |                     |                         |
| + MaptoIneligible               | <True>           |             |                     |                         |
| + CreateApplication             |                  | <Global>    | <False>             | <Commit>                |

6. Expand the **AccountVerification** component. Under each component is its implementation, along with the set of its interfaces and references. Some qualifiers are disabled. Some are not. Only those qualifiers that are not disabled can be modified.

The following table shows the qualifiers that determine the Quality of Service (QoS) for the components. [More](#)

All Qualifiers
Reliability Qualifiers
Activity Session Qualifiers
Security Qualifiers
Options...

| Component                       | Reliability      |             |                     |              |
|---------------------------------|------------------|-------------|---------------------|--------------|
|                                 | Join transaction | Transaction | Suspend transaction | Asynchronous |
| ▶ AccountVerificationExport     |                  |             |                     |              |
| ◀ AccountVerification           |                  |             |                     |              |
| Implementation                  | Global           |             |                     |              |
| ◀ References                    |                  |             |                     |              |
| CreditCheckServicePartner       |                  |             | False               | Commit       |
| GenerateDeclinePartner          |                  |             | False               | Commit       |
| RequestMoreDocumentatio         |                  |             | False               | Commit       |
| CreditRiskAssessmentPartn       |                  |             | False               | Commit       |
| FinalApplicationReviewPart      |                  |             | False               | Commit       |
| RecordDeclinedApplication       |                  |             | False               | Commit       |
| DetermineApplicationEligibl     |                  |             | False               | Commit       |
| MaptoIneligiblePartner          |                  |             | False               | Commit       |
| RecordIneligibleApplicatio      |                  |             | False               | Commit       |
| ◀ Interfaces                    |                  |             |                     |              |
| ◀ AccountVerification           | False            |             |                     |              |
| InputCriterion                  |                  |             |                     |              |
| ▶ DetermineApplicantEligibility | <True>           | <Global>    |                     |              |
| ▶ RecordIneligibleApplication   | <True>           |             |                     |              |
| ▶ CreditScoreServiceImport      | <False>          |             |                     |              |

The table currently shows the following qualifiers:

- **Join transaction**

This qualifier determines whether the target service can join a client-propagated transaction. You can apply the qualifier to all of its interfaces, to an individual interface, or to an individual operation on an interface.



### Information

If **Join transaction** is set to `true`, it joins any transaction that a client propagates, if the implementation runs in a global transaction. Otherwise, the transaction does not join any transaction that a client propagates.

- **Transaction**

This qualifier is set on an implementation. It determines the logical unit of work that the component processing runs.



## Information

The transaction qualifier can be set to the following settings:

- **Global**: A global transaction is a unit of work in a distributed transaction processing environment in which multiple resource managers can be enlisted. If the client propagates a transaction and a component is configured to join that transaction, then this component is run under that transaction. In other words, resources that the component updates are committed when the client commits the transaction. If either of these conditions is missing, the component runs under a new global transaction that is committed when the operation completes.
- **Local (default)**: This setting is required when a component implementation is not capable of participating in a two-phase commit protocol. Sometimes ACID coordination of multiple resource managers is not needed. In such scenarios, running business logic in a local transaction does better than in a global transaction. Therefore, the component runs in a local transaction, even if a transaction is propagated to the hosting runtime environment. Local transactions are not propagated to services that the component starts. If an activity session is active, the local transaction joins it. If an activity session is not currently active, one is started automatically.
- **Any**: If a global transaction context is propagated from the client, the runtime environment dispatches methods from the component in a global transaction; otherwise, the runtime environment establishes a local transaction environment.
- **Local application**: If this option is selected, the component processing occurs within a WebSphere local transaction containment that the application manages. It is the responsibility of the application to commit or to roll back transactions. (Selecting this option is equivalent to setting the transaction descriptor to `Not Supported` and setting the `LocalTranResolver` to `application` for an enterprise bean.)

---

- **Suspend transaction**

This qualifier, which is set on a reference, determines whether the client propagates the transaction to the referenced component.



## Information

By default, this value is `false`, which means that synchronous invocations run completely within any client global transaction. Otherwise, when the value is `true`, synchronous invocations occur outside any client global transaction.

---

- **Asynchronous invocation**

This qualifier determines whether asynchronous invocations can occur as part of a client transaction.



## Information

The asynchronous invocation qualifier can have these settings:

- **Commit:** The message is sent only if the transaction is committed. Asynchronous invocations with the reference are transacted as a part of any client global transaction or extended local transaction (that is, where the **Transaction** value qualifier for the client implementation is local).
- **Call** (default): Asynchronous invocations with the reference occur immediately. This property is logically equivalent to suspending the current transaction and then starting the target service.



## Questions

**What do the current qualifiers on the AccountVerification SCA component mean?**

- On the implementation, the **Transaction** qualifier is set to `Global`. If that client propagated the transaction, and `AccountVerification` is configured to join it, then the `AccountVerification` component runs under the same transaction as its client.
- On the interface, the **Join transaction** qualifier is set to `false`. `AccountVerification` does not join any transaction that its client propagates. When you consider that this qualifier is `false`, but the qualifier on the implementation is `Global`, the two qualifiers together mean that `AccountVerification` runs under its own, new global transaction.
- On the references, the **Suspend transaction** and **Asynchronous invocation** qualifiers are all set to `false` and `Commit`. `AccountVerification` does not propagate its transaction to any of its references, and it sends its messages to its references only if the transaction is committed.

7. For the **DetermineApplicationEligibilityPartner** reference on the `AccountVerification` component, set the **Suspend transaction** qualifier to `true`. In this way, the `DetermineApplicantEligibility` component, which has a Java implementation, joins the same transaction as `AccountVerification`.

| References                                          |  |                    |                     |  |
|-----------------------------------------------------|--|--------------------|---------------------|--|
| <code>RecordIneligibleApplicationPartner</code>     |  | <code>False</code> | <code>Commit</code> |  |
| <code>DetermineApplicationEligibilityPartner</code> |  | <code>True</code>  | <code>Commit</code> |  |
| <code>RecordDeclinedApplicationPartner</code>       |  | <code>False</code> | <code>Commit</code> |  |
| <code>FinalApplicationReviewPartner</code>          |  | <code>False</code> | <code>Commit</code> |  |
| <code>CreditRiskAssessmentPartner</code>            |  | <code>False</code> | <code>Commit</code> |  |
| <code>GenerateDeclinePartner</code>                 |  | <code>False</code> | <code>Commit</code> |  |
| <code>CreditCheckServicePartner</code>              |  | <code>False</code> | <code>Commit</code> |  |
| <code>MapToIneligiblePartner</code>                 |  | <code>False</code> | <code>Commit</code> |  |
| <code>RequestMoreDocumentationPartner</code>        |  | <code>False</code> | <code>Commit</code> |  |

You cannot set the **Asynchronous invocation** qualifier on the `DetermineApplicationEligibilityPartner` reference to `Call`. It generates a problem

because the invocation is not asynchronous: it is a synchronous call to an SCA component with a Java implementation.

- 8. Save your work.
- 



### Note

Remember that setting these qualifiers in this lab does not affect the testing environment in any way. Changes to these settings are intended to provide you with an example on how to change transaction handling with SCA components. No test can be done in this simulated environment that adequately shows the effects of transaction management.

---

## **Part 3: Examine the transaction boundaries for activities in long-running business processes**

Microflow business processes are run in a single thread and a single transaction. However, long-running (interruptible) business processes can be multithreaded and can contain multiple transactions.

The following list describes the transaction boundaries of a long-running process. You can influence transaction boundaries for BPEL activities by using the transactional behavior attribute. However, the Business Flow Manager can add or remove transaction boundaries at any time in the runtime environment.

In general, a transaction boundary is needed in the following situations:

- When waiting for an external request, that is, upon reaching a receive activity or a receive choice activity for which a corresponding request was not received
- When scheduling a timer for a wait activity
- When starting a service asynchronously with an invoke activity
- When starting a human task activity

In addition, Business Flow Manager introduces transaction boundaries in the following situations at run time. Your process design cannot rely on these boundaries as they can be overridden during process navigation.

- When a fault is raised during process navigation
- Before and after the invoke activity is started, which starts a service synchronously (and the service does not participate in the transaction of the process)
- When propagating lifecycle operations to child processes; for example, when a parent process is suspended, its child processes are suspended in subsequent transactions
- When the process instance is deleted automatically upon completion of the process
- When trying to recover from a failure that causes the rollback of a transaction, which spans a series of activities



### Hint

If you need guaranteed transaction boundaries, factor out the business logic that must be run in a single transaction into a microflow, and start it as a subprocess. The logic of a microflow is always run in a single transaction. Invocations in a microflow must be synchronous. Asynchronous calls within a microflow (such as human task invocations) are not allowed.

When you model a business process, you can suggest transaction boundaries for invoke, snippet, and human task activities by changing the transactional behavior attribute of the corresponding activity. If an invoke activity calls a synchronous service that does not participate in the current transaction, then the transactional behavior attribute is ignored. In this case, a transaction boundary always exists before the invoke activity is started, and after the invoke activity completes.

The transactional behavior attribute can take one of the following values:

- **Commit before**

The current transaction is committed, and a new transaction is started. The activity with this attribute value becomes the first activity of the new transaction.

- **Commit after**

The activity participates in the current transaction. After the activity completes successfully, the transaction is committed, and a new one is started. A new transaction is started for each immediately following activity, and each subsequent activity becomes the first activity of one of these new transactions.

- **Participates**

The activity participates in the current transaction. More transaction boundaries are not set, either before or after the activity.

In the following situations, and depending on the values of their settings of the transactional behavior attributes, this setting allows the transaction to continue with the navigation of the following activities.

- If the invoke activity starts the service asynchronously, the arrival of the response message triggers a new transaction. The transaction is short because it commits immediately after the status of the invoke activity is updated.
- In a sequence of human task activities, two transactions are needed for each human task activity, one to activate the human task activity and another to complete the human task activity. If you change the setting to Participates, you can reduce the number of transactions to one for each human task activity. The completion of the previous human task activity, and the activation of the following activity, are done in the same transaction.

- **Requires own**

The activity runs in its own transaction. As a result, the current transaction is committed before the activity starts, and a new transaction starts after this activity completes.

You can also determine whether the transaction that initiates the process is committed after certain receive activities complete. Change the transactional behavior attribute of the receive activity, or

the receive action of the receive-choice (pick) activity. For initiating receive and receive-choice activities, the attribute can take one of the following values:

- **Commit after**

After the activity completes successfully, the transaction that initiates the process is committed, and a new one is started. This setting is useful if you start the process instance with a synchronous API call.

- **Participates**

The transaction that initiates the process continues after the activity completes. This setting is required if you want to start the process instance with the `initiateAndClaimFirst` API. With this API, you can create a process instance and immediately claim the first human task.

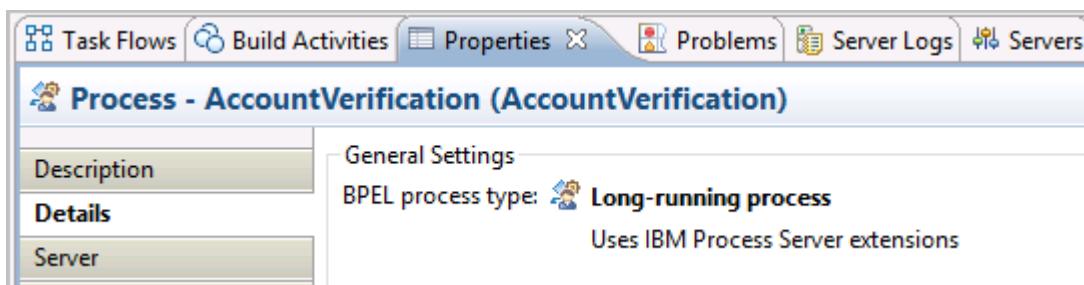
If your process starts another BPEL process, ensure that the corresponding invoke activity is not part of the transaction that initiates the process. You can achieve this result by setting the transaction behavior attribute in one of the following ways:

- Set the attribute of the initiating receive or receive choice activity to Commit after.
- Set the attribute of the invoke activity to Commit before or Requires own.

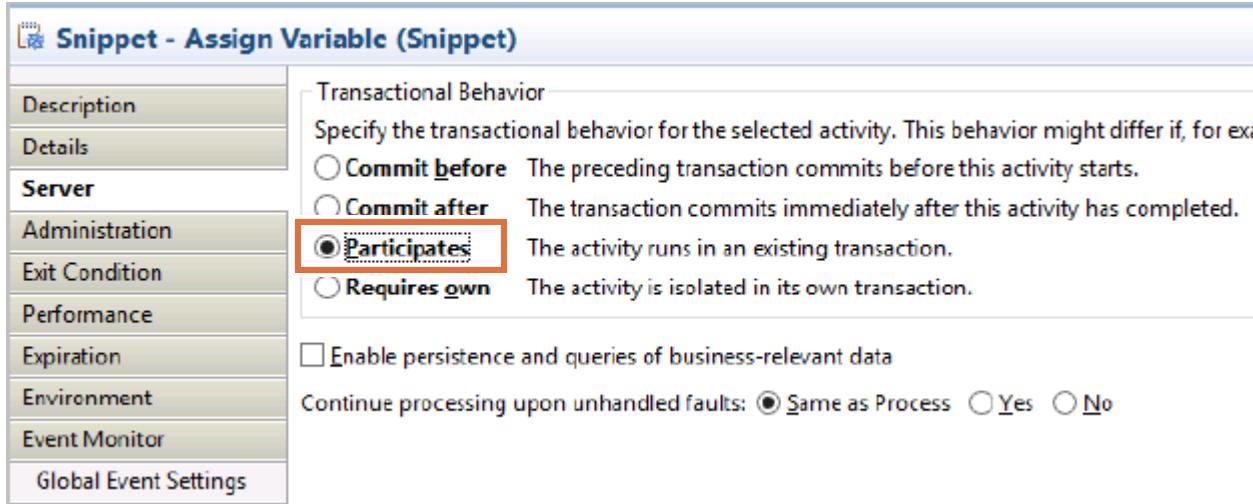
In long-running processes, you can change transaction boundaries. In most cases, performance can be improved by reducing the number of transaction boundaries. However, the optimal number of transaction boundaries can be found only through performance testing.

To identify the transactional behavior for activities in the long-running **AccountVerification** business process:

- 1. Verify that **AccountVerification** is a long-running business process. You cannot modify transactional behavior for activities in a microflow. Microflows always run in a single transaction.
  - a. In the **Business Integration** view, expand **FoundationModule > Integration Logic > BPEL Processes > AccountVerification**.
  - b. Double-click **AccountVerification** to open it in the business process editor.
  - c. In the business process editor, click the **AccountVerification** icon at the top of the tray.
  - d. Switch to the **Properties** view.
  - e. In the **Properties** view, select the **Details** tab. Verify that the **BPEL Process Type** is set to **Long-running process**.

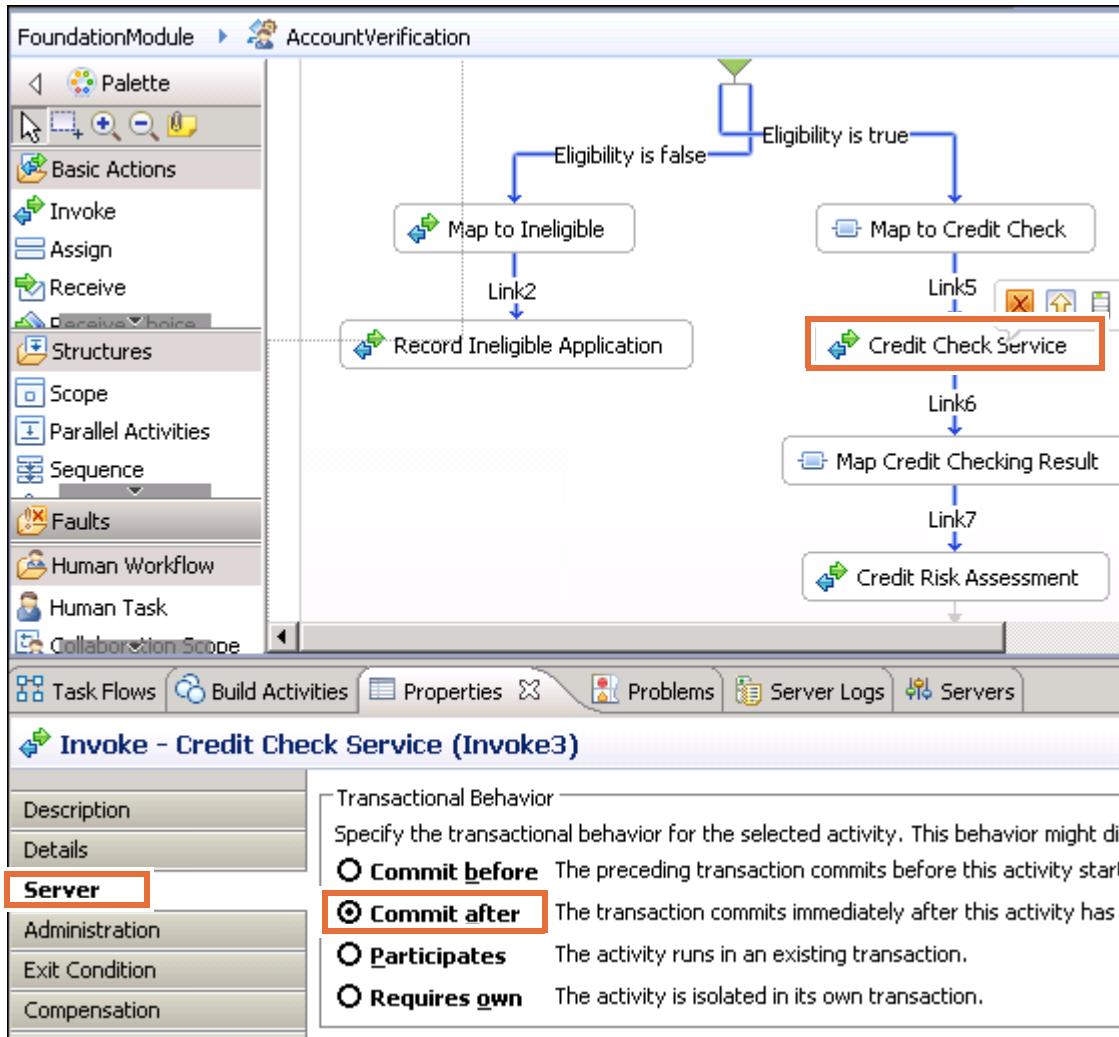


- 2. Review the **Transactional Behavior** settings for the **Assign Variable** snippet and the **Credit Check Service** invoke activity. User settings for human tasks, invoke activities, and snippets can affect transaction boundaries.
- a. Click the **Assign Variable** snippet activity.
  - b. Switch to the **Properties** view.
  - c. Select the **Server** tab. The **Transactional Behavior** setting is set to **Participates** by default.



- d. Select the **Credit Check Service** invoke activity.
- e. Switch to the **Properties** view.
- f. Select the **Server** tab.

- g. The **Transactional Behavior** setting must be set to **Commit After**. The **Transactional Behavior** setting on invoke activities is set to **Commit After** by default, while it is set to **Participates** on snippet activities.



3. As time allows, examine the transaction settings for other activities.

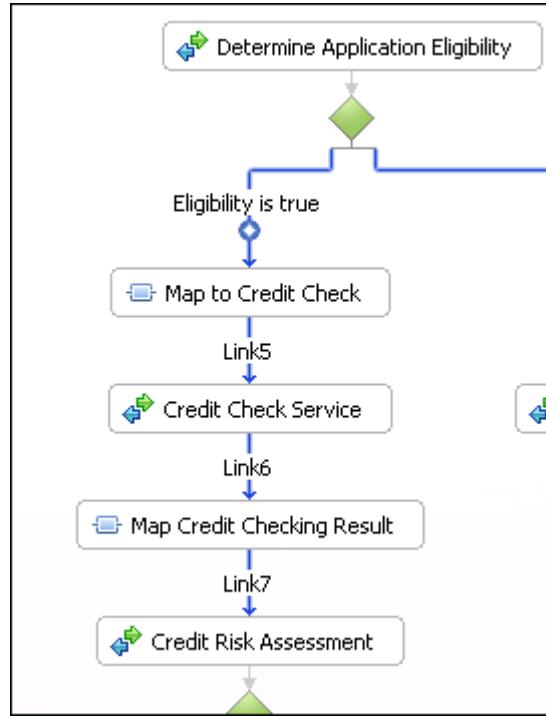
#### **Part 4: Use a scope in a long-running business process to manage persistence**

Long-running business processes differ from microflows in the ways that they handle persistence. Microflows are single-threaded business processes that run in a single transaction, and are not persisted without being committed. However, in a long-running business process, you are able to manage and control the persistence and commit tasks at various stages of the process.

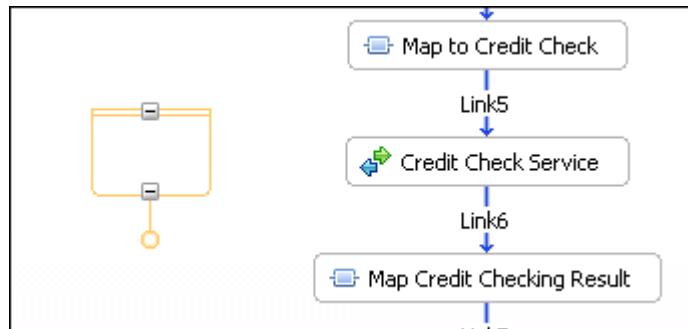
A **scope** is a container in a business process for grouping, compartmentalizing, or organizing similar activities. This fact also extends to persistence. Activities that are part of the same scope can be persisted as part of the scope to visually control transaction boundaries. You can also

persist them to maintain the process at various stages of completion to provide crash recovery or compensation checkpoints.

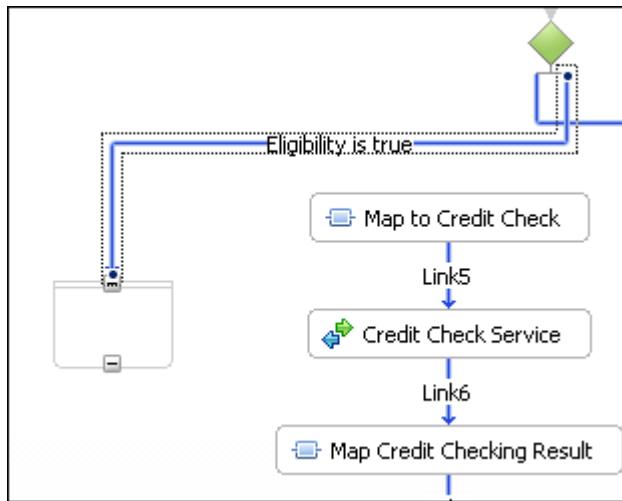
- 1. In the `AccountVerification` business process, three logical steps exist in the process that manage the credit check: `Map to Credit Check`, the `Credit Check Service` invocation, and `Map Credit Checking Result`.



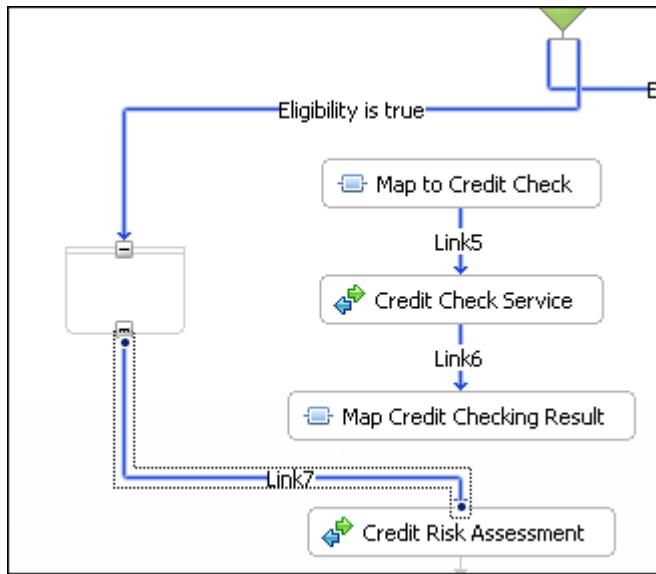
- 2. In the business process editor palette, expand **Structures** and click **Scope**. Drop a scope into the `AccountVerification_Flow` structure, next to these credit checking activities.



- 3. Select the wire that connects the application eligibility gateway, labeled `Eligibility is true`, and reconnect it to the top of the new scope.



- 4. Select the wire that connects the `Map Credit Checking Result` activity to the `Credit Risk Assessment` activity, and reconnect it to the bottom of the scope.



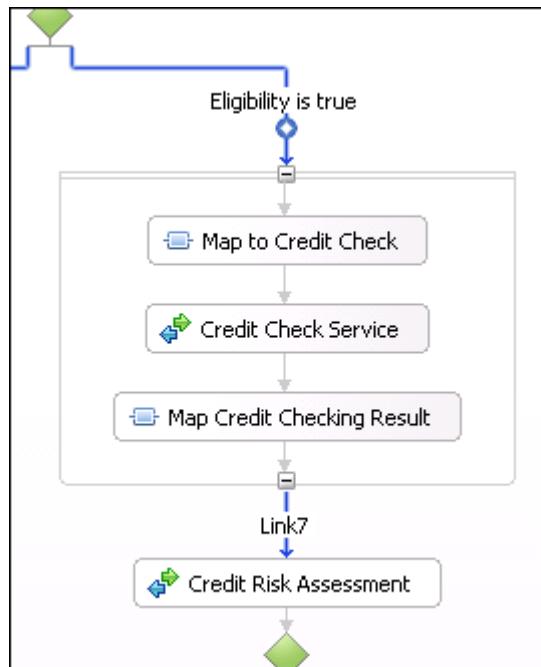
- 5. Drag the `Map to Credit Check` activity into the new scope. The new scope does not organize activities in parallel, like `AccountVerification_Flow`, so the flow arrows are applied sequentially.



### Note

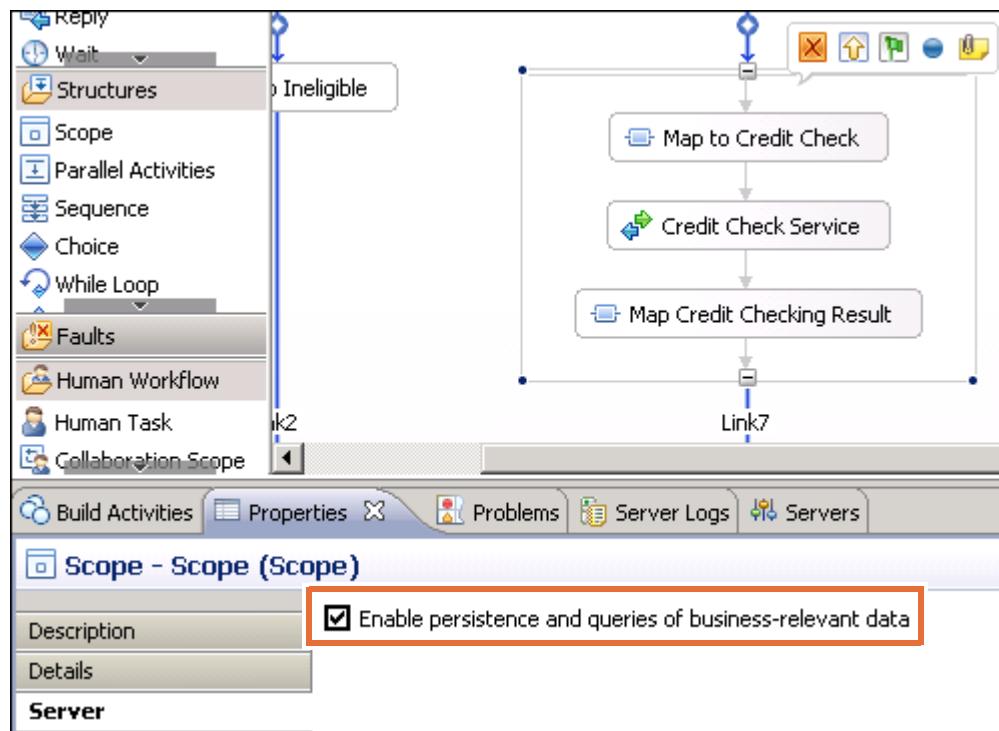
If you do not see the activity after dropping into the new scope, then right-click the scope and choose **Expand All Activities**.

- \_\_\_ 6. Drag the Credit Check Service invocation activity into the new scope. Guidelines assist you so that you can place it beneath the Map to Credit Check activity. Because activities in this scope are sequential, they are linked automatically.
- \_\_\_ 7. Drag Map Credit Checking Result into the scope and place it beneath the Credit Check Service. The contents of the scope are set.
- \_\_\_ 8. Right-click an available space in the diagram and click **Align Parallel Activities Contents Automatically**.

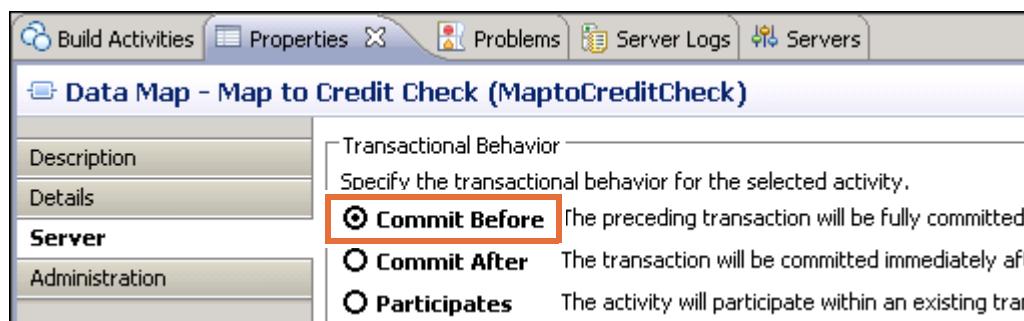


- \_\_\_ 9. Configure persistence settings on the new scope.
  - \_\_\_ a. Select the scope in the business process diagram.
  - \_\_\_ b. Select the **Properties** view.
  - \_\_\_ c. Click the **Server** tab.

- \_\_\_ d. Make sure that the **Enable persistence and queries of business-relevant data** setting is checked.

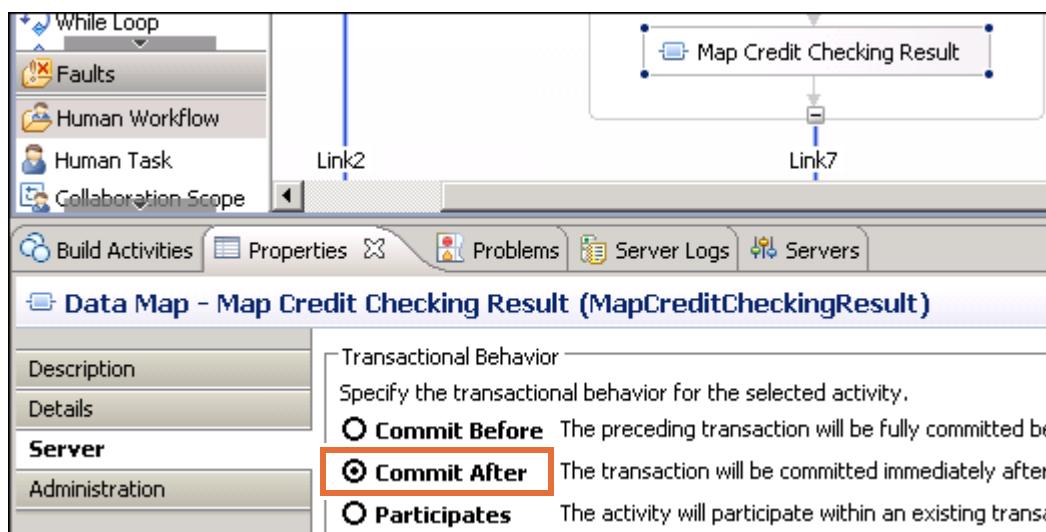


- \_\_\_ 10. Change the transaction processing settings so that the transaction is committed before processing begins in the scope and after the activities in the scope are complete.
- Select the **Map to Credit Check** activity.
  - Select the **Properties** view.
  - Click the **Server** tab.
  - Change the transactional behavior property to **Commit Before**. This setting ensures that the transaction is fully committed before the processing of the activities in the scope commences.



- \_\_\_ e. Select the **Map Credit Checking Result** activity.

- f. Change the transactional behavior property to **Commit After**. This setting ensures that the activities in the scope are fully committed before processing resumes.



- 11. Save your work.  
— 12. Close IBM Integration Designer. Click **File > Exit**, or click **X** in the upper-right corner.

## End of exercise

# Exercise 8. Creating flexible business processes

## Estimated time

01:30

## Overview

This exercise introduces business process flexibility concepts.

## Objectives

After completing this exercise, you should be able to:

- Use the Business Process Choreographer Explorer client to dynamically change the execution path of a running process instance
- Implement a selector component to choose between multiple service invocations
- Test a selector component in the IBM Integration Designer integrated test client

## Introduction

Many different process flexibility mechanisms are available in IBM Integration Designer. This exercise introduces two of them: selectors and dynamic navigation. Selector components allow dynamic service invocations that are based on date and time criteria. Dynamic navigation allows users or administrators to skip and jump activities in a running process instance.

## Requirements

To complete the exercises for this course, you need a lab environment that includes the following items: the exercise support files, Mozilla Firefox, IBM Integration Designer V8.5.7, IBM DB2 V10.1, RFHUTIL, WebSphere MQ V8.0, and the IBM Process Server V8.5.7 test environment.

## Exercise instructions

# Implementing a selector component

Routing communication from the client and target applications through a selector has several advantages:

- It decouples the client application from the target so that the target can change over time without affecting the client.
- The implementation of the selector and the destination can be independent and as such, either can be changed as necessary without affecting the other.
- Target implementations can be added without changing the client or requiring a restart.

Selectors can be classified into three groups:

- Generic date-range-based selectors

These selectors make their invocation decision that is based on which range a date lies in. Extra ranges can be added in the runtime environment; targets can be changed, and the resulting modifications can be re-exported back to the tool.

- Generic non-date-range-based selectors

These selections are based on other data types and conditions. Examples include StringSingleton (does an incoming string match a specific string?) and IntegerRange (does an incoming integer fit into a particular range?).

- Custom selectors

If the other selectors that are described do not meet your needs, then you can write your own algorithm.

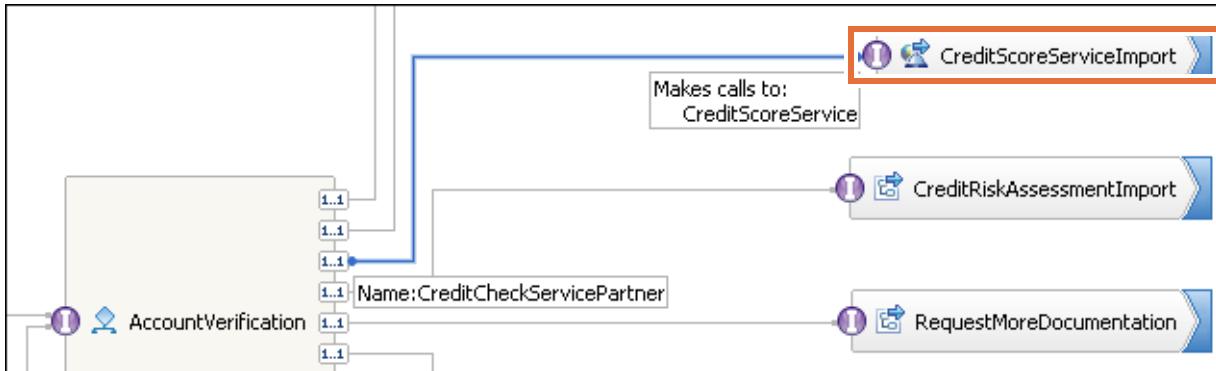
### **Part 1: Adding a selector**

In this portion of the exercise, you implement a selector component that invokes one of three possible credit score service providers that are based on date and time selection criteria.

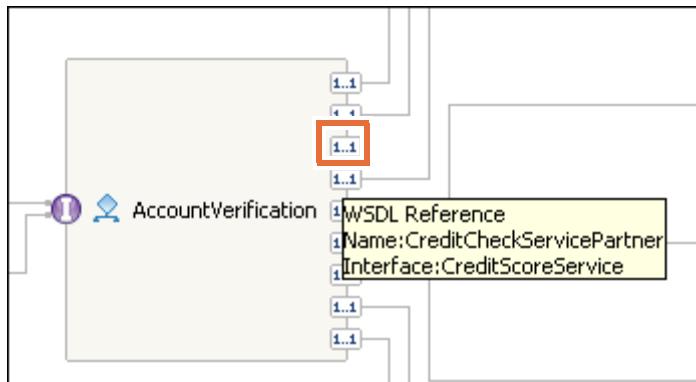
To implement a selector component:

- 1. Open the Exercise 8 workspace.
  - a. On your desktop, open the folder that is labeled **Exercise Shortcuts**.
  - b. Double-click the shortcut that is labeled **Exercise 8** to start IBM Integration Designer.
  - c. Wait for the workspace build to complete; this build process might take a few minutes.
  - d. Close the **Getting Started** tab.
- 2. Open the **FoundationModule** assembly diagram.
  - a. In the **Business Integration** view, expand **FoundationModule**.
  - b. Double-click **Assembly Diagram** to open the assembly editor.

- \_\_\_ 3. Delete the **CreditScoreServiceImport** component and the **CreditCheckServicePartner** reference.
    - \_\_\_ a. On the assembly diagram, select the **CreditScoreServiceImport** component.



- \_\_ b. Press **Delete** to remove the import.
  - \_\_ c. Click **Yes** when you get the **Confirm Delete** window.
  - \_\_ d. On the **AccountVerification** process component, select the **CreditCheckServicePartner** reference. Hovering the mouse pointer over the reference reveals the reference name and interface.

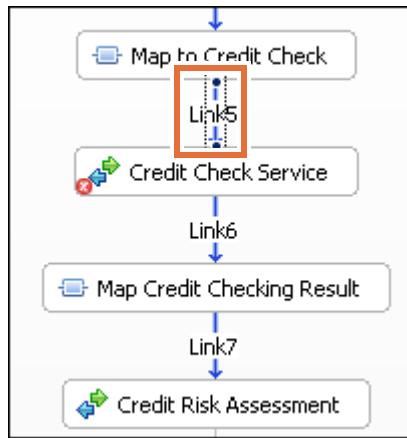


- \_\_\_ e. Press **Delete** to remove the reference partner.
  - \_\_\_ f. Save your changes. You can ignore any errors in the Problems view.

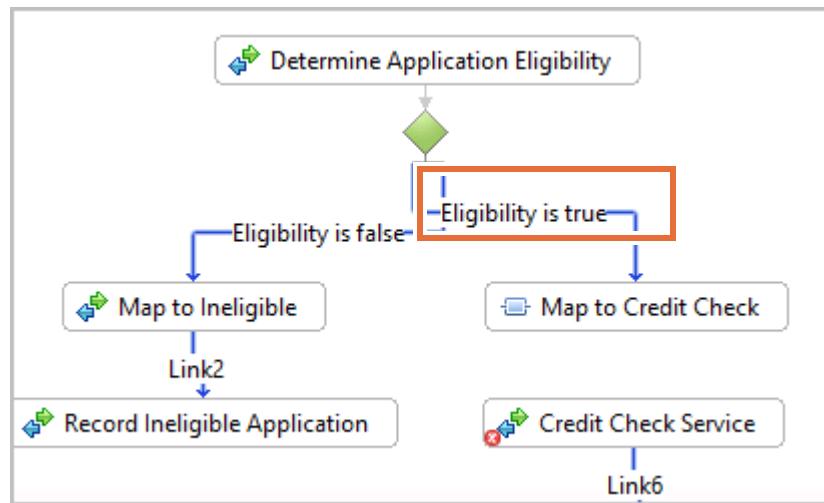
\_\_\_ 4. Synchronize the changes that are made on the assembly diagram with the business process implementation.

  - \_\_\_ a. On the **FoundationModule** assembly diagram, right-click the **AccountVerification** process component.
  - \_\_\_ b. From the menu, choose **Synchronize Interfaces and References > to Implementation**.
  - \_\_\_ c. In the **Confirm Synchronize to Implementation** window, click **Yes**. Clicking Yes removes the CreditCheckServicePartner from the Reference Partners section of the tray in the BPEL editor.

- 5. Delete the **Map to Credit Check** and **Map Credit Checking Result** data map activities from the **AccountVerification** business process. Transforming the data before it reaches the credit score providers is done in a separate module. The reason for this delete becomes clearer later in the exercise.
- a. On the **FoundationModule** assembly diagram, double-click the **AccountVerification** process component to open the BPEL editor.
  - b. In the **AccountVerification** process, select the link between the **Map to Credit Check** activity and the **Credit Check Service** activity. This link is **Link5** on the diagram.

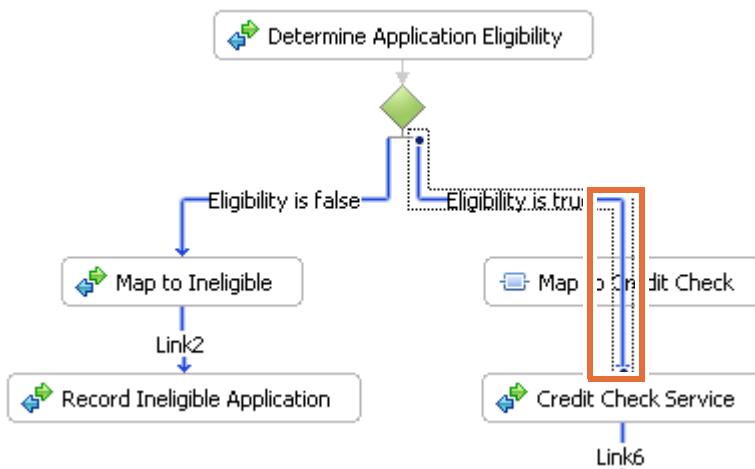


- c. Press **Delete** to remove it.
- d. Click the **Eligibility is true** link between **Determine Application Eligibility** and **Map to Credit Check**.

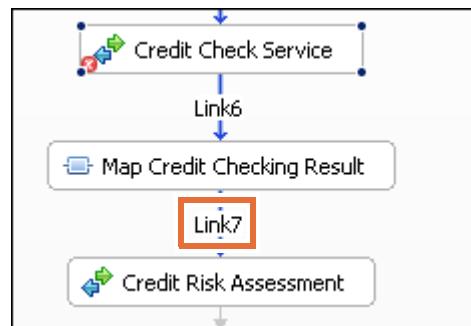


- e. Hover the mouse pointer over the point at which the link connects to the **Map to Credit Check** activity. (The pointer becomes two crossed, double-headed arrows that are also known as a “crosshair”).

- \_\_\_ f. When the pointer becomes the crossed, double-headed arrows, click and drag the endpoint of the link onto the **Credit Check Service** invoke activity. This action moves the link so that it is now between **Determine Application Eligibility** and **Credit Check Service**.

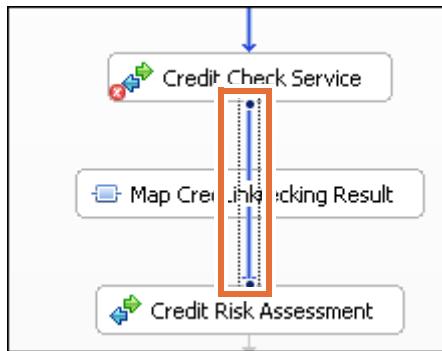


- \_\_\_ g. *Delete the **Map to Credit Check** activity.*
- \_\_\_ h. Save your changes.
- \_\_\_ i. *Delete the link between **Map Credit Checking Result** and the **Credit Risk Assessment** invoke activity. It is **Link7** on the diagram.*

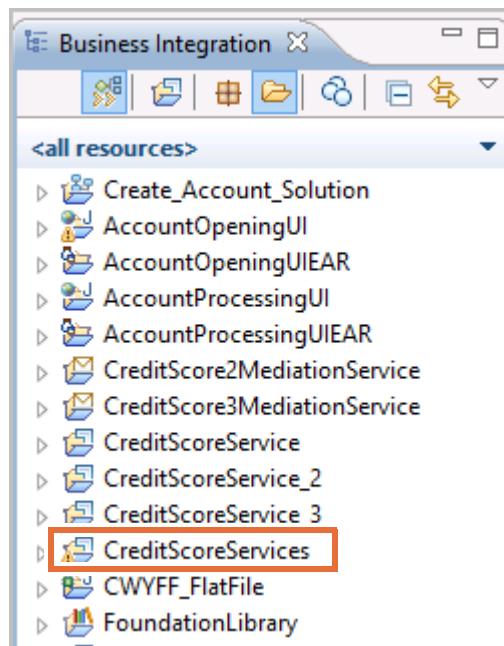


- \_\_\_ j. Select the link between the **Credit Check Service** invoke activity and the **Map Credit Checking Result** activity. It is **Link6** on the diagram.
- \_\_\_ k. Hover the mouse pointer over the point at which **Link6** connects to the **Map Credit Checking Result** activity. (The pointer becomes two crossed, double-headed arrows that are also known as a “crosshair”).

- I. When the pointer becomes the crossed, double-headed arrows, click and drag the endpoint of the link onto the **Credit Risk Assessment** invoke activity. This action moves the link so that it is now between **Credit Check Service** and **Credit Risk Assessment**.

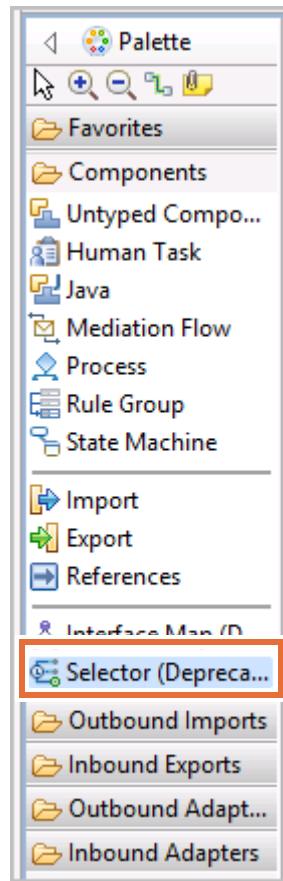


- m. Delete the **Map Credit Checking Result** data map activity.
- n. Save your changes and close the **AccountVerification** BPEL editor tab. Continue to ignore any errors in the Problems view.
- 6. Add a selector component that is named **CreditScoreProviderSelector** to the **CreditScoreServices** assembly diagram.
- a. In the **Business Integration** view, expand **CreditScoreServices** and double-click **Assembly Diagram**. Verify that you select **CreditScoreServices** and not **CreditScoreService**.



- b. In the **CreditScoreServices** assembly editor, in the **Palette** expand the **Components** drawer.

- \_\_ c. Click **Selector** and then click an empty space on the assembly diagram to add the component.



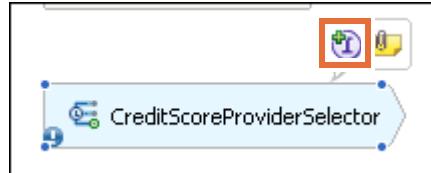
- \_\_ d. Make sure that the **Selector** component is selected, and switch to the **Description** tab in the **Properties** view.
- \_\_ e. Place the cursor in the **Name** field and change the value to:  
CreditScoreProviderSelector

|                |               |                             |
|----------------|---------------|-----------------------------|
| Description    | Name:         | CreditScoreProviderSelector |
| Details        | Display name: | CreditScoreProviderSelector |
| Implementation | Folder:       |                             |
| All Qualifiers | Description:  |                             |

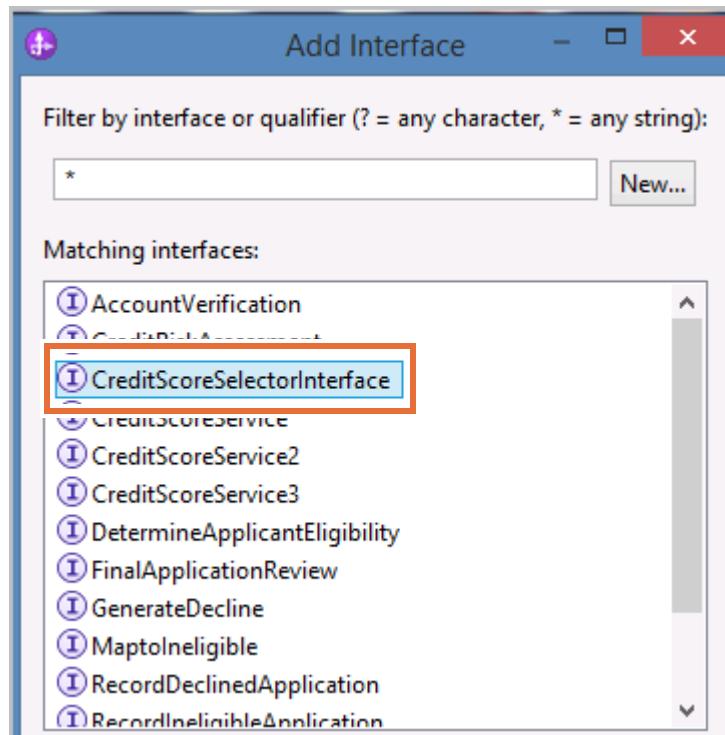
- \_\_ f. Save your changes.

- 7. Add **CreditScoreSelectorInterface** to the **CreditScoreProviderSelector** component. This interface was created for you to save time. As time allows, feel free to examine the interface definition in FoundationLibrary.

- a. On the **CreditScoreServices** assembly diagram, click **CreditScoreProviderSelector**.
- b. In the pop-up action bar, click the **Add Interface** icon.



- c. In the **Add Interface** window, choose **CreditScoreSelectorInterface**.

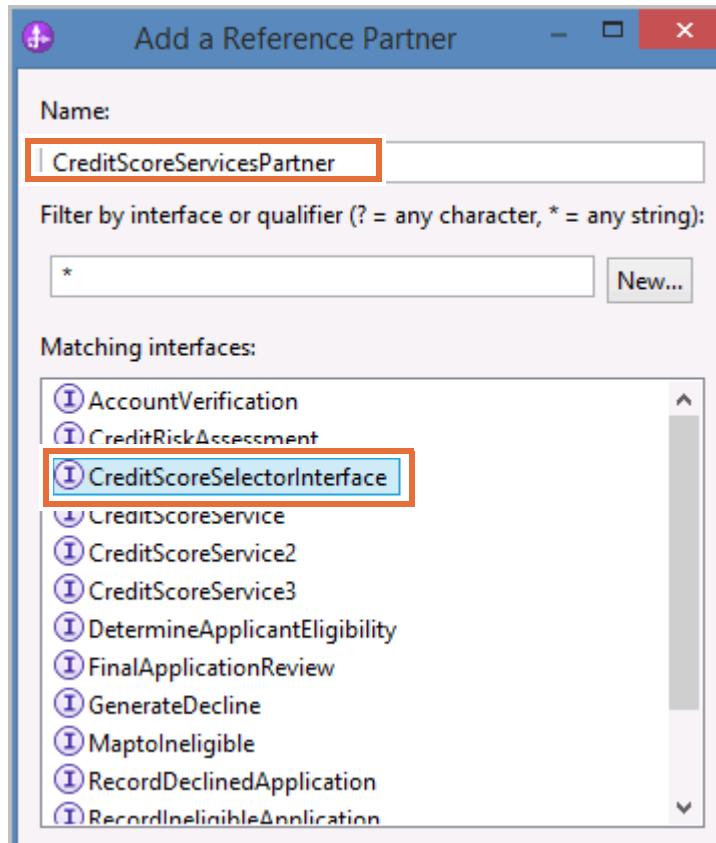


- d. Click **OK**.
  - e. Save your changes.
- 8. Generate an SCA bound export component for the Selector so that the AccountVerification business process can call it.
- a. On the **CreditScoreServices** assembly diagram, right-click the **CreditScoreProviderSelector** component.
  - b. Choose **Generate Export > SCA Binding** from the menu.



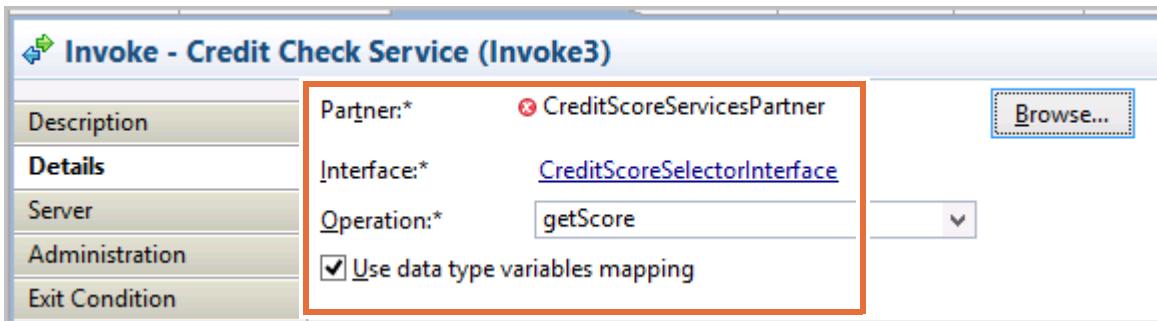
- c. Save your changes.

- 9. Add a reference partner that is named **CreditScoreServicesPartner** to the **AccountVerification** business process. This reference partner is used to call the **CreditScoreProviderSelectorExport** interface in the **CreditScoreServices** module.
  - a. In the **Business Integration** view, expand **FoundationModule > Integration Logic > BPEL Processes > AccountVerification**.
  - b. Double-click **AccountVerification** to open the process in the BPEL editor (if the AccountVerification BPEL process tab is still open from the previous portion of the exercise, then you can switch to that tab).
  - c. In the **Reference Partners** section of the tray, click the **Add a Reference Partner** icon (the green plus symbol).
  - d. In the **Add a Reference Partner** window, complete the following actions:
    - Enter **CreditScoreServicesPartner** in the **Name** field.
    - In the **Matching Interface** window, select **CreditScoreSelectorInterface**.



- e. Click **OK** to add the partner to the tray.
- f. Save your changes.
- 10. In the **AccountVerification** business process, change the implementation of the **Credit Check Service** invoke activity so it uses the **CreditScoreServicesPartner** to call the **CreditScoreSelectorInterface** on **CreditScoreProviderSelectorExport**.
  - a. Right-click the **Credit Check Service** activity and choose **Show In > Properties view** from the menu.

- \_\_ b. Switch to the **Details** tab in the **Properties** view.
- \_\_ c. To the right of the **Partner** field, click **Browse**.
- \_\_ d. In the **Select a Partner** window, select **CreditScoreServicesPartner**.
- \_\_ e. Click **OK**.
- \_\_ f. Verify that the **Interface** field is set to **CreditScoreSelectorInterface**, that the **Operation** field is set to **getScore**, and that **Use data type variables mapping** is checked.

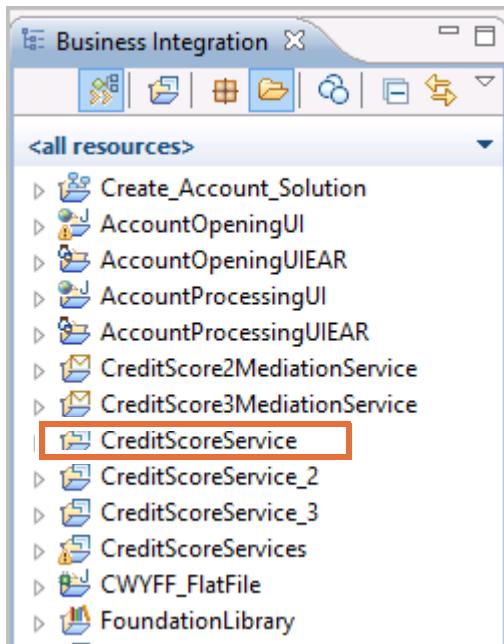


- \_\_ g. In the data type variables table, in the **Inputs** row, click the **none** link in the **Read from Variable** column.
- \_\_ h. Select **CustomerApplicationVariable** from the list.
- \_\_ i. In the data type variables table, in the **Outputs** row, click the **none** link in the **Store into Variable** column.
- \_\_ j. Select **CustomerApplicationVariable** from the list.

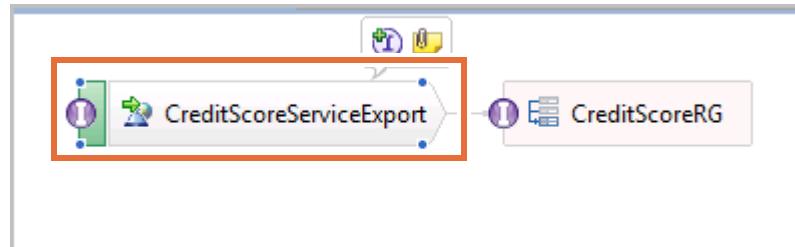
|         | Name   | Type                | Read from Variable          |
|---------|--------|---------------------|-----------------------------|
| Inputs  | Input  | CustomerApplication | CustomerApplicationVariable |
|         | Name   | Type                | Store into Variable         |
| Outputs | Output | CustomerApplication | CustomerApplicationVariable |

- \_\_ k. Save your changes.
- \_\_ l. Close the AccountVerification BPEL editor. Continue to ignore any errors in the Problems view.

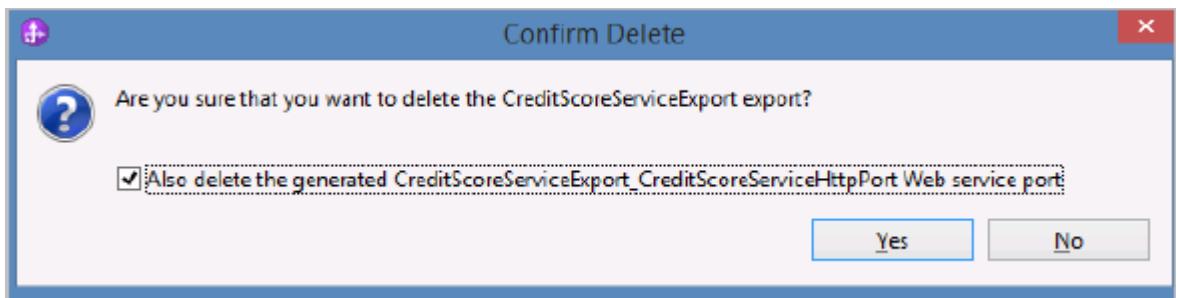
- \_\_ 11. Regenerate the **CreditScoreService** export component.
  - \_\_ a. In the **Business Integration** view, expand **CreditScoreService** and double-click **Assembly Diagram**. Verify that this time you select **CreditScoreService** and not **CreditScoreServices**.



- \_\_ b. On the assembly diagram, select the **CreditScoreServiceExport** component.

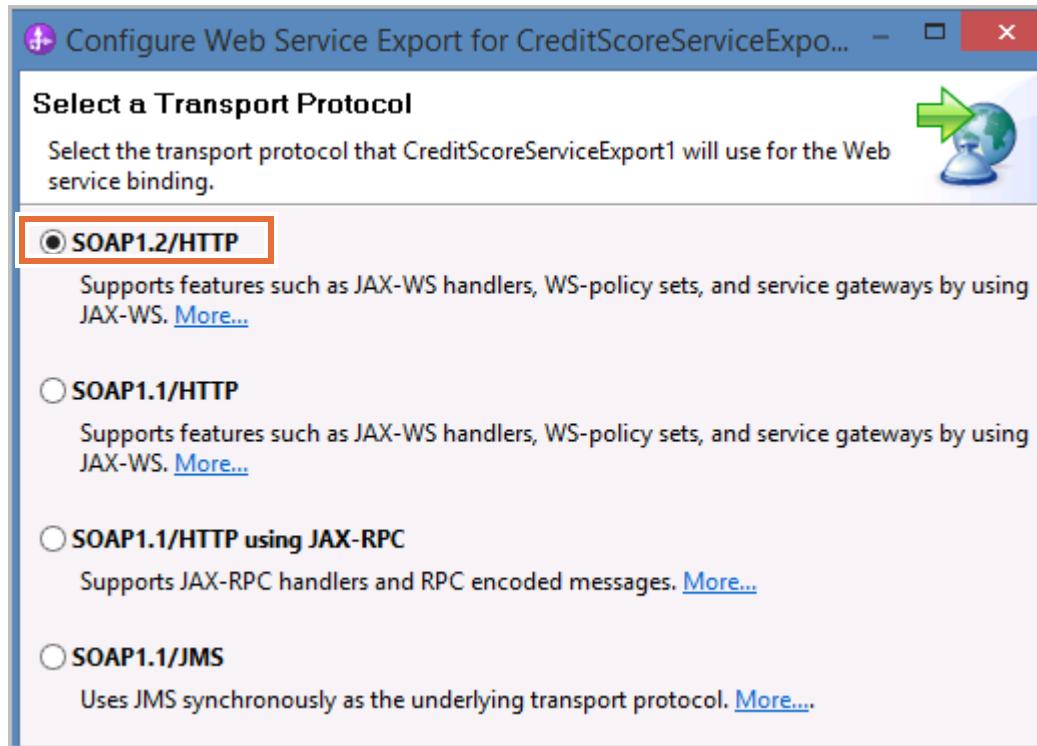


- \_\_ c. Press **Delete** to remove the component.
- \_\_ d. In the **Confirm Delete** window, check the option to delete the generated web service port and click **Yes**.

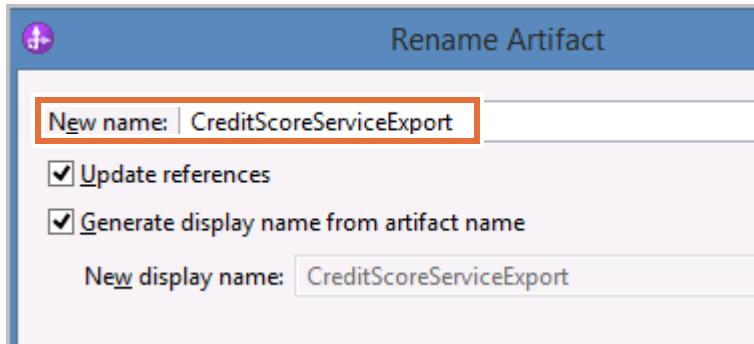


- \_\_ e. In the **CreditScoreService - Assembly Diagram**, right-click **CreditScoreRG** and select **Generate Export > Web Service Binding**.

- \_\_ f. In the **Select a Transport Protocol** window, select **SOAP1.2/HTTP**, and click **Finish**.

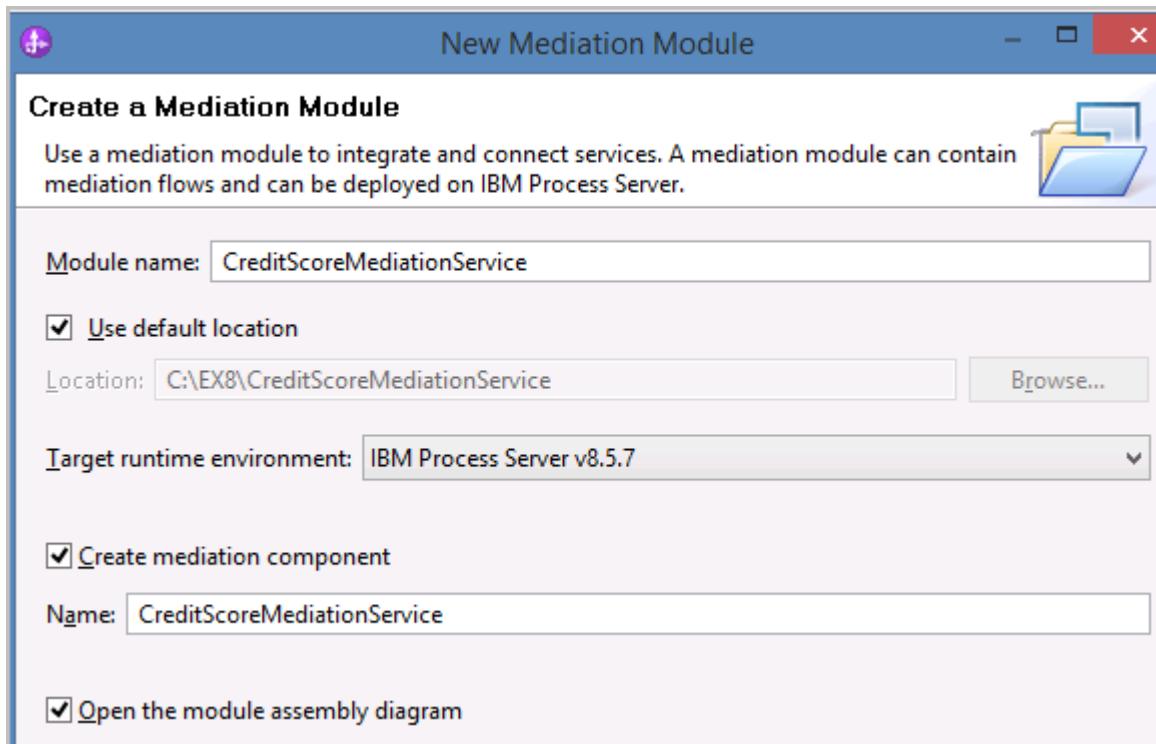


- \_\_ g. Save your changes by Selecting **File** and **Save All**.
- \_\_ h. With **CreditScoreServiceExport1** selected, switch to the **Description** tab in the **Properties** view.
- \_\_ i. Place the cursor in the **Name** field and press **Alt+Shift+R** to refactor it.
- \_\_ j. In the **Rename Artifact** window, change the **New name** to:  
CreditScoreServiceExport



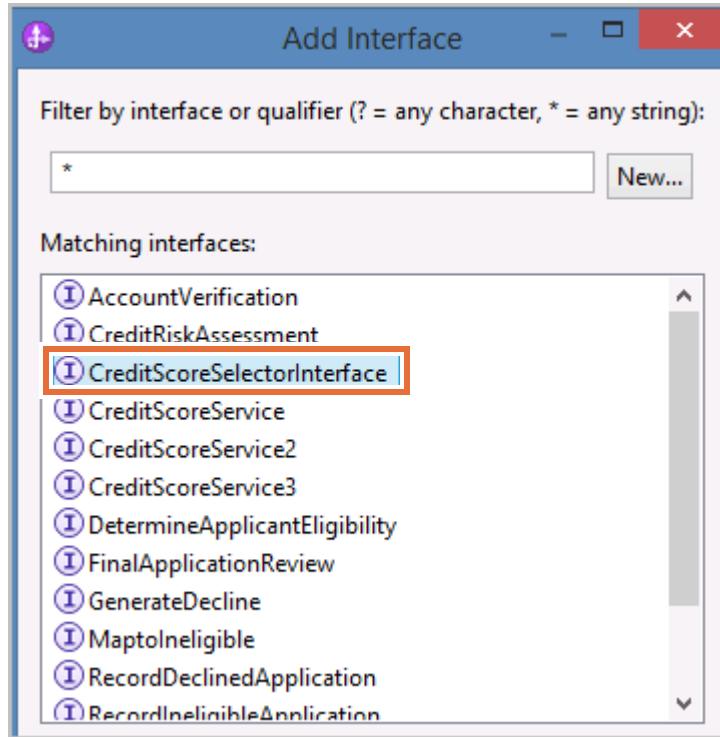
- \_\_ k. Accept the remaining default options and click **Refactor**.
- \_\_ l. Right-click the newly created  
**CreditScoreServiceExport\_CreditScoreServiceHttpPort** under **CreditScoreService > Web Service Ports** and click **Copy**.
- \_\_ m. Right-click **FoundationLibrary** and click **Paste**.
- \_\_ n. If the **Resource Exists** window is displayed, click **Yes** to override.

- 12. Create a mediation module that reconciles the interface mismatch between the selector component and the CreditScoreService web service. In previous releases, you can reconcile this mismatch by using an interface mapping component. Now that the interface map is deprecated, you must use a mediation module and an XSLT mapping primitive for this function. To save time, mediations that reconcile the selector interface and the interfaces of CreditScoreService2 and CreditScoreService3 are created for you.
- a. From the menu options, choose **File > New > Mediation Module**.
  - b. At the **Create a Mediation Module** window, enter the following information.
    - In the **Module Name** field, enter: **CreditScoreMediationService**
    - In the **Target Runtime Environment** field, choose **IBM Process Server v8.5.7** from the list.



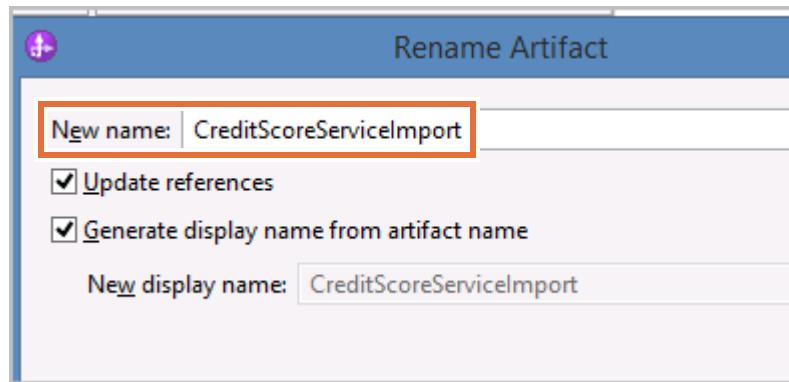
- c. Accept the remaining default options and click **Next**.
  - d. At the **Select the Required Libraries** window, select **FoundationLibrary**, and click **Finish**.
- 13. Add the **CreditScoreSelectorInterface** to the **CreditScoreMediationService** flow component.
- a. On the **CreditScoreMediationService** assembly diagram, right-click **CreditScoreMediationService** and choose **Add > Interface** from the menu.

- \_\_\_ b. In the **Add Interface** window, select **CreditScoreSelectorInterface**.

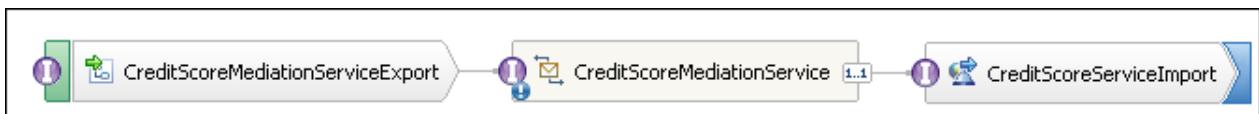


- \_\_\_ c. Click **OK**.
- \_\_\_ 14. Generate an SCA bound export component for the **CreditScoreMediationService** flow component.
- \_\_\_ a. On the **CreditScoreMediationService** assembly diagram, right-click **CreditScoreMediationService** and choose **Generate Export > SCA Binding** from the menu.
- \_\_\_ b. Save your changes.
- \_\_\_ 15. Add a web service import component to the **CreditScoreMediationService** assembly diagram that calls **CreditScoreService**.
- \_\_\_ a. In the **Business Integration** view, expand **FoundationLibrary > Web Service Ports**.
- \_\_\_ b. Drag **CreditScoreServiceExport\_CreditScoreServiceHttpPort** onto the **CreditScoreMediationService** assembly diagram.
- \_\_\_ c. Save your changes.
- \_\_\_ 16. Change the **Name** of the import component to **CreditScoreServiceImport**.
- \_\_\_ a. With **CreditScoreServiceImport1** selected, switch to the **Description** tab in the **Properties** view.
- \_\_\_ b. Place the cursor in the **Name** field and press **Alt+Shift+R** to refactor it.

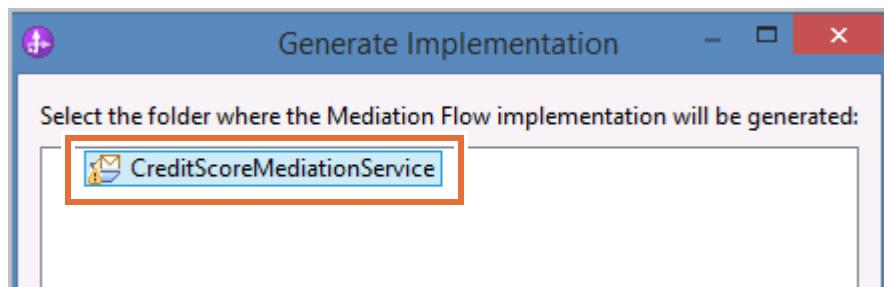
- \_\_\_ c. In the **Rename Artifact** window, change the **New name** to:  
CreditScoreServiceImport



- \_\_\_ d. Accept the remaining default options and click **Refactor**.
- \_\_\_ 17. Wire the **CreditScoreServiceImport** component to the **CreditScoreMediationService** flow component.
- \_\_\_ a. Select the **CreditScoreMediationService** flow component.
  - \_\_\_ b. Hover the mouse pointer over the right side of the component until you see the orange, circular “**Add wire**” handle.
  - \_\_\_ c. Click and drag the handle onto the **CreditScoreServiceImport** interface.
  - \_\_\_ d. At the **Add Wire** window, click **OK** to add a matching reference to the flow component.
  - \_\_\_ e. Save your changes.



- \_\_\_ 18. Generate the **CreditScoreMediationService** implementation by using the **Operation Map** template.
- \_\_\_ a. On the **CreditScoreMediationService** assembly diagram, double-click the **CreditScoreMediationService** flow component.
  - \_\_\_ b. Click **Yes** when you are prompted to implement the component.
  - \_\_\_ c. In the **Generate Implementation** window, leave the **CreditScoreMediationService** folder selected.

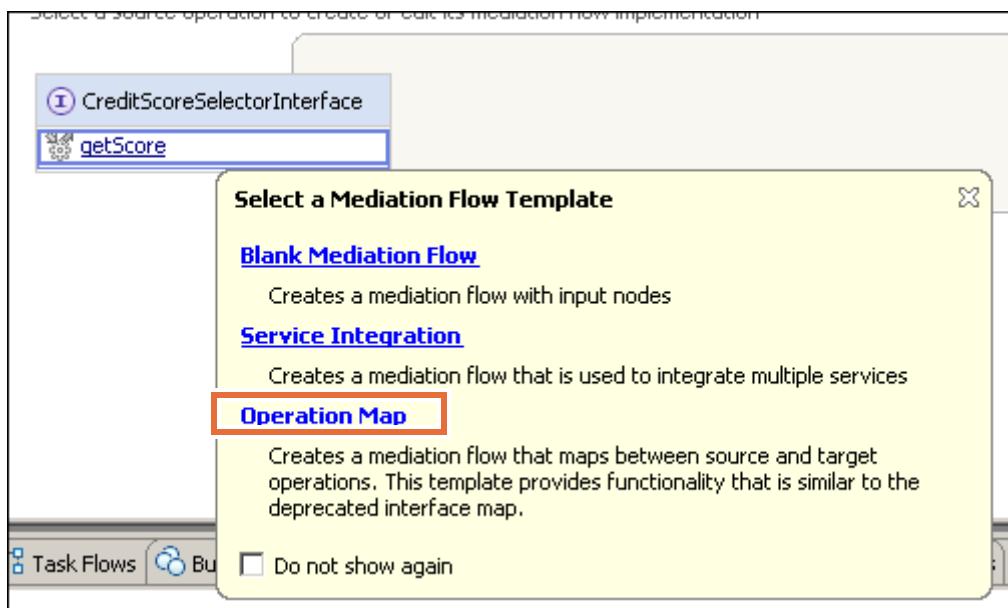


- \_\_\_ d. Click **OK**.

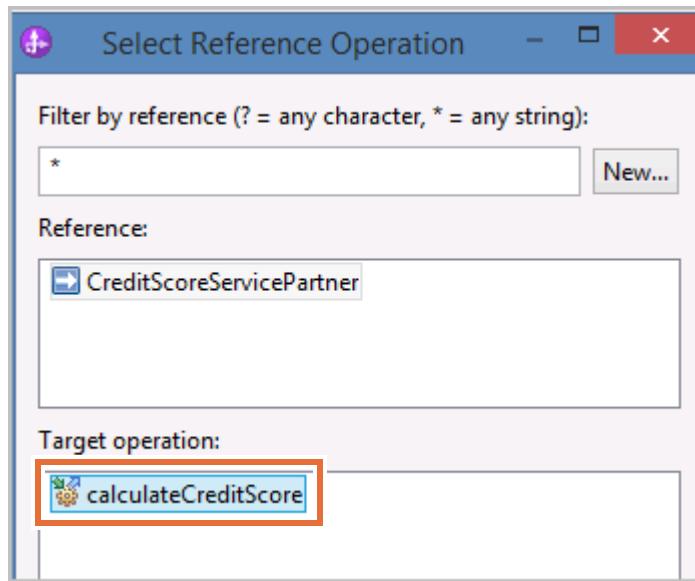
- \_\_ e. In the **Tip** window, select the **Do not show tips** check box and close the window.
- \_\_ f. On the **Overview** tab, in the **Operation connections** section of the diagram, click the **getScore** operation link on the **CreditScoreSelectorInterface**.



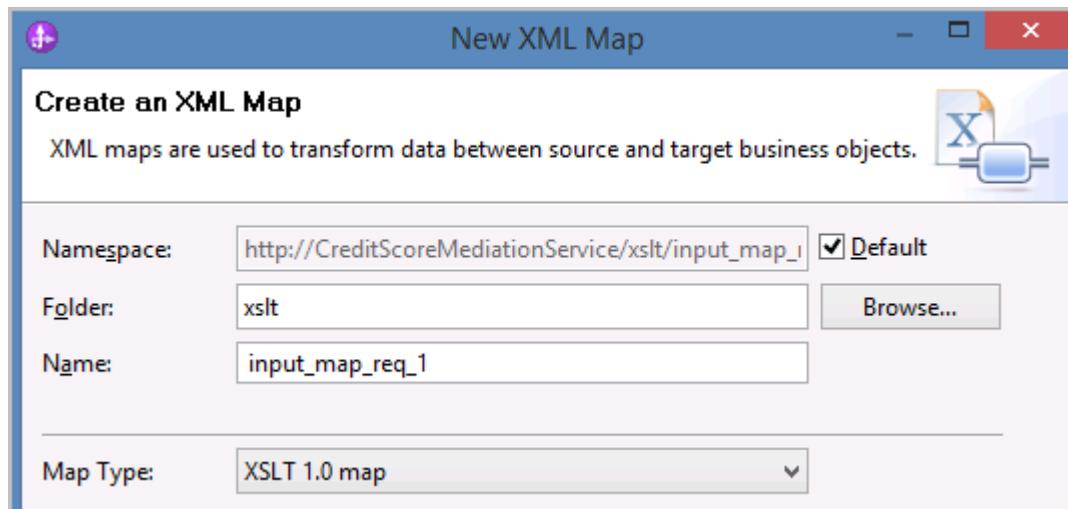
- \_\_ g. In the **Select a Mediation Flow Template** window, click the **Operation Map** link.



- \_\_ h. In the **Select Reference Operation** window, leave the **calculateCreditScore** operation selected.

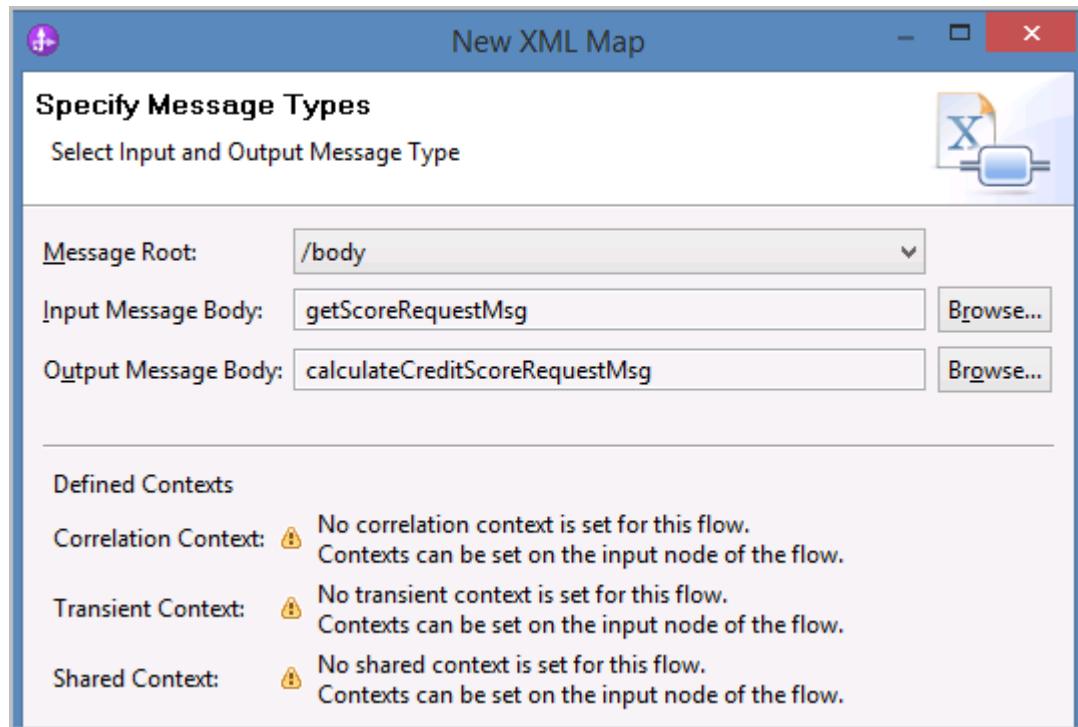


- \_\_ i. Click **OK**. It generates a request flow and a response flow, each of which uses an XSL transformation primitive to map the **CustomerApplication** business object to a **CreditCheckRequest** business object.
- \_\_ j. Save your changes.
- \_\_ 19. Implement the **input\_map** data map on the **CreditScoreMediationService Request** flow.
- \_\_ a. On the **Request** tab of the **CreditScoreMediationService** flow diagram, double-click the **input\_map** XSL primitive.
  - \_\_ b. At the **Create an XML Map** window, accept the default values for **Name** and **Folder**.



- \_\_ c. Click **Next**.

- \_\_\_ d. At the **Specify Message Types** window, accept the default values. You can ignore any warnings at the bottom of the window.



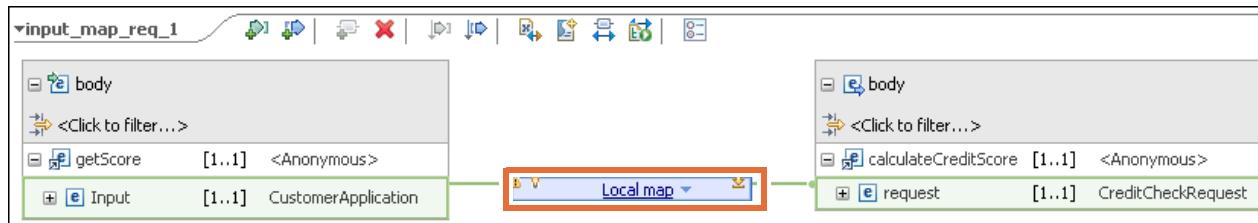
### Note

The Message Root field identifies the part of the message that you want to map in the XML mapping editor.

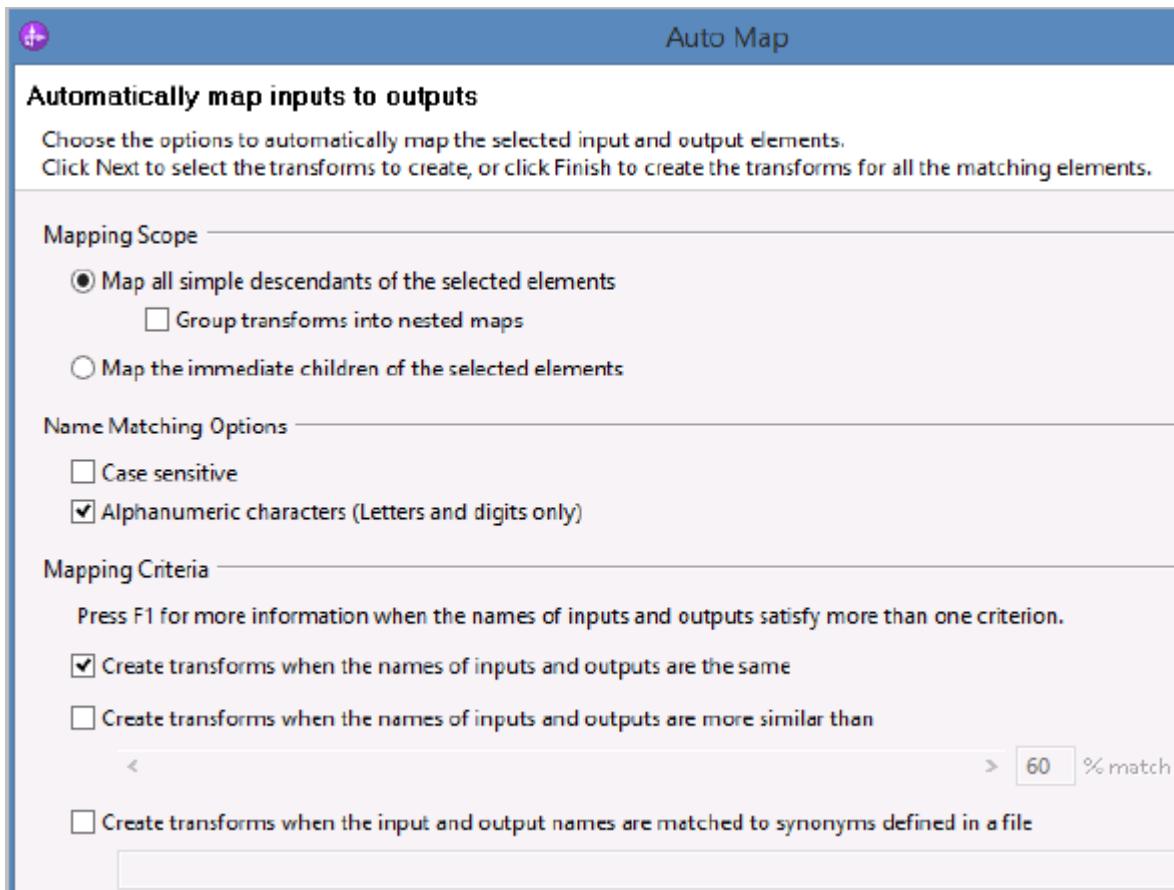
- / (root) transforms the entire message and is used to map SOAP attachments.
- /body transforms the message body.
- /headers transforms the message headers.
- /context transforms the message context.

- \_\_\_ e. Click **Finish**.
- \_\_\_ 20. Create a local map and connect the fields in the source object to the fields in the target object by using Move transformations.
- \_\_\_ a. In the mapping editor, click the + symbols to expand the elements in the input operation and the output operation.
- \_\_\_ b. Hover the mouse pointer over the `CustomerApplication` input (on the left side of the diagram).

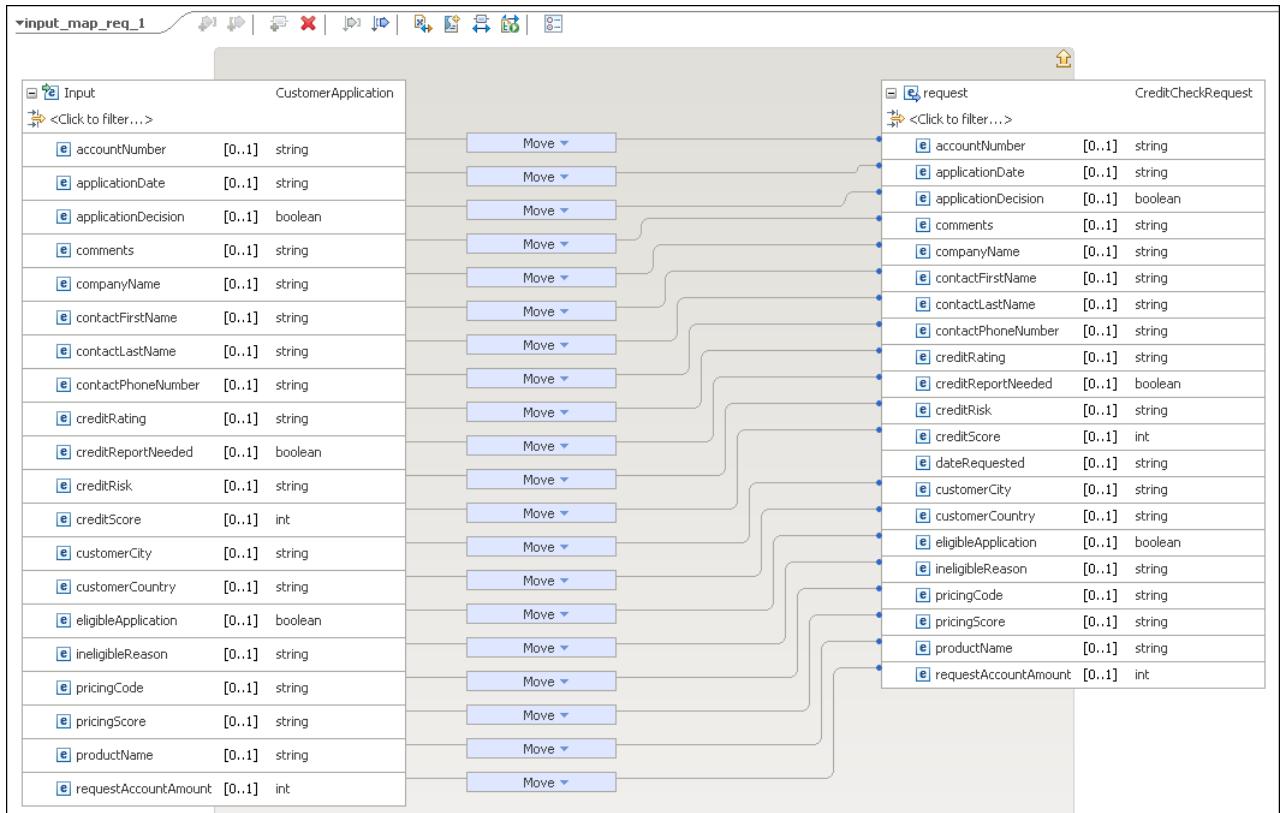
- \_\_ c. Drag the orange, circular handle from the CustomerApplication input to the CreditCheckRequest output. It creates a **Local map** transformation.



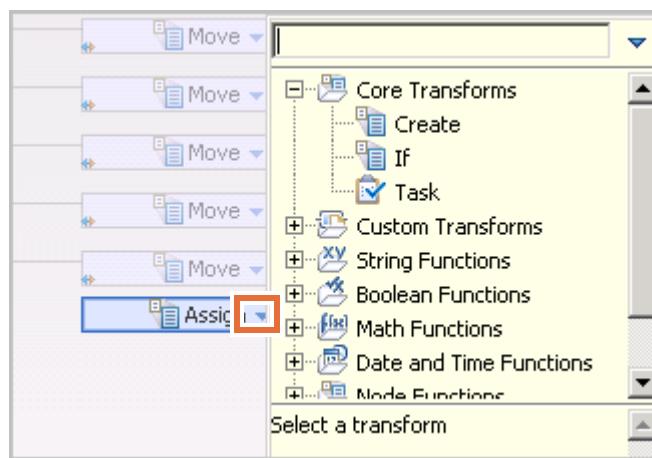
- \_\_ d. Click the **Local map** transformation.  
 \_\_ e. In the local map, click the **Auto map input to output** icon (the fifth icon from the right). In the **Auto map** window, accept the default values and click **Finish**.



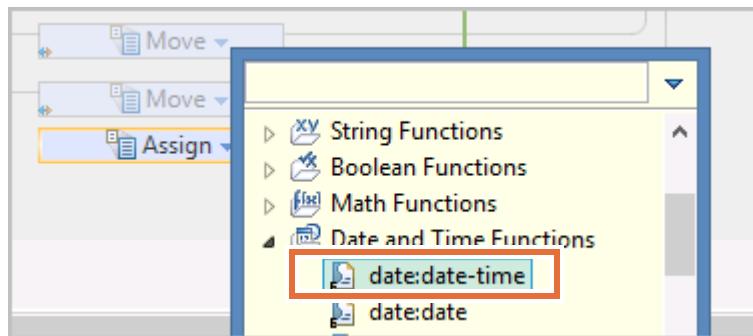
- \_\_\_ f. It creates a series of Move transformations from the attributes in the source object to the attributes in the target object.



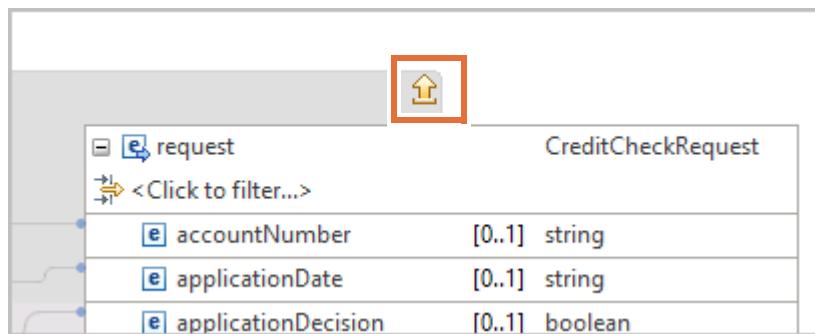
- \_\_\_ 21. Create an **Assign** transformation that places the current date and time in the `dateRequested` attribute of the `CreditCheckRequest` target.
- \_\_\_ a. Right-click `dateRequested` in the `CreditCheckRequest` target and choose **Create Assign** from the menu.
- \_\_\_ b. Click the down arrow in the **Assign** transformation to open the window.



- \_\_\_ c. Expand **Date and Time Functions** and select **date:date-time**.



- \_\_\_ d. At the top of the local map, click the **Up a level** icon (the up arrow) to exit.



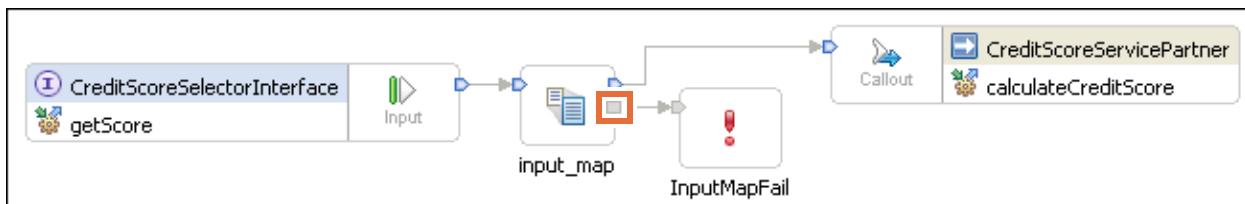
- \_\_\_ e. Save your changes and close the data map editor.

\_\_\_ 22. (Optional) Delete the sticky note.

- \_\_\_ a. Right-click the **TODO** sticky note and choose **Delete** from the menu to remove it from the diagram.
- \_\_\_ b. Save your changes to the CreditScoreMediationService Request flow diagram. Continue to ignore any errors in the Problems view.

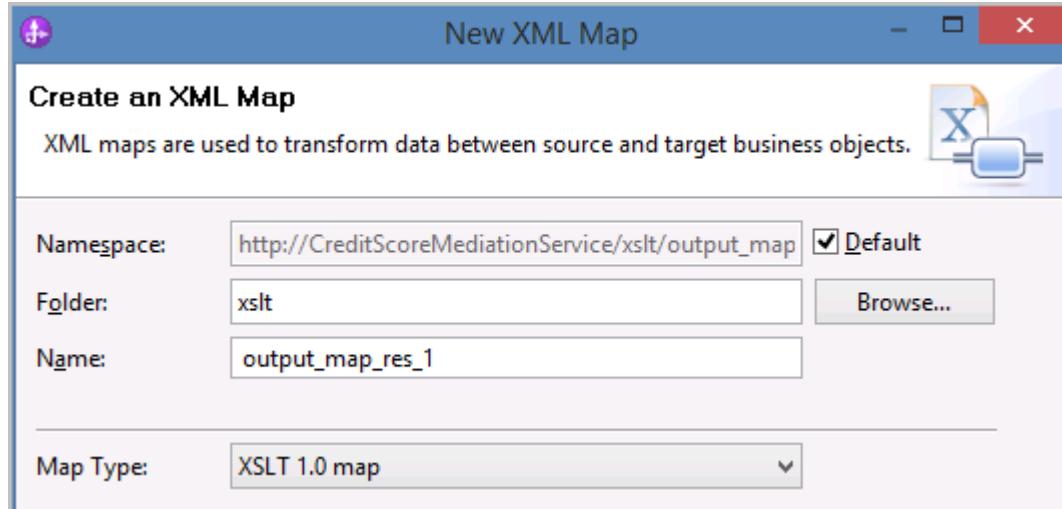
\_\_\_ 23. Add a **Fail** node to the **Request** flow that throws an exception for a failure in the mapping primitive.

- \_\_\_ a. In the **Palette**, expand the **Error Handling** drawer and click **Fail**.
- \_\_\_ b. Click the flow diagram just below **input\_map** to add it to the diagram.
- \_\_\_ c. Change the default display name from **Fail1** to: **InputMapFail**
- \_\_\_ d. Hover the mouse pointer over the **Fail** terminal on the **input\_map** primitive.
- \_\_\_ e. Drag the orange, circular **Add wire** handle to the **in** terminal on the **Fail** primitive.
- \_\_\_ f. Save your changes. The Request flow looks like the following screen capture:

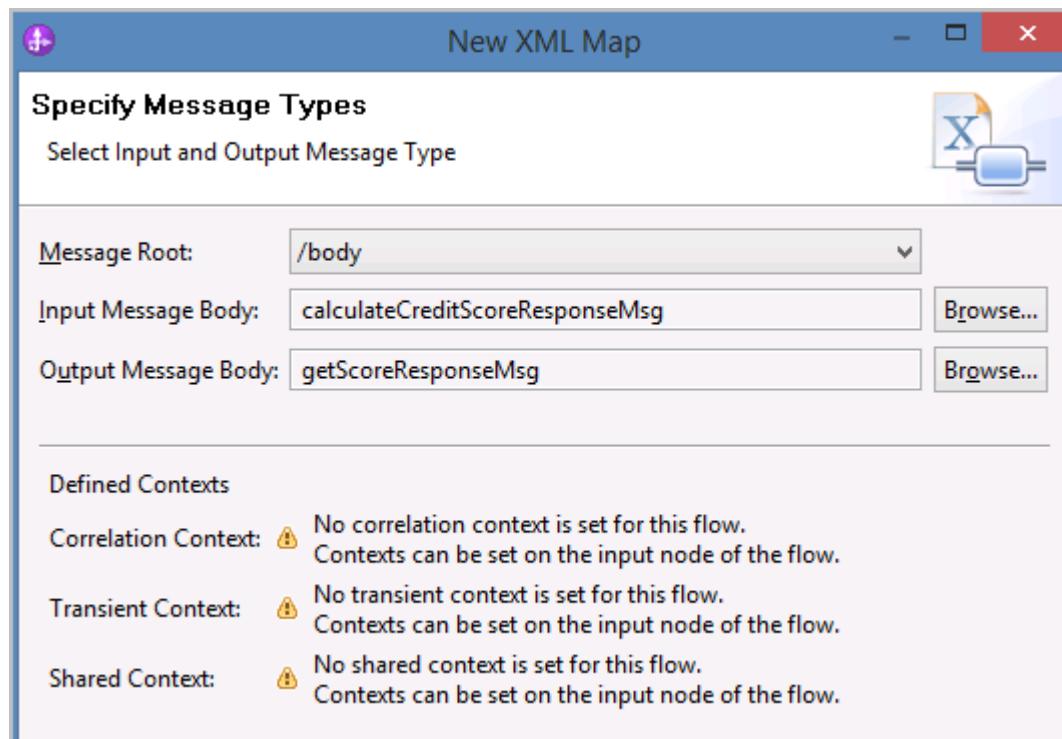


\_\_\_ 24. Switch to the **Response** flow tab.

- 25. Implement the **output\_map** data map on the **CreditScoreMediationService Response** flow.
- a. On the **CreditScoreMediationService Response** flow diagram, double-click **output\_map** to open the mapping editor.
  - b. At the **Create an XML Map** window, accept the default options.

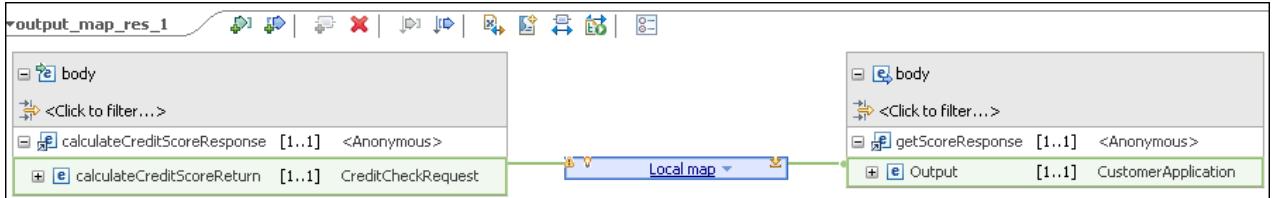


- c. Click **Next**.
- d. At the **Specify Message Types** window, accept the default options. You can ignore any warnings at the bottom of the window.

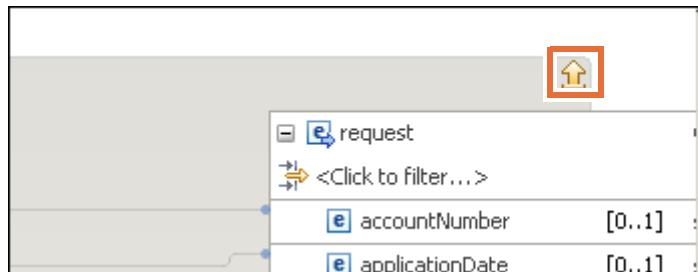


- e. Click **Finish**.

- 26. Create a local map and connect the fields in the source object to the fields in the target object by using Move transformations.
- a. In the mapping editor, click the + symbols to expand the elements in the input operation and the output operation.
  - b. Hover the mouse pointer over the `CreditCheckRequest` input (on the left side of the diagram).
  - c. Drag the orange, circular handle from the `CreditCheckRequest` input to the `CustomerApplication` output. It creates a **Local map** transformation.



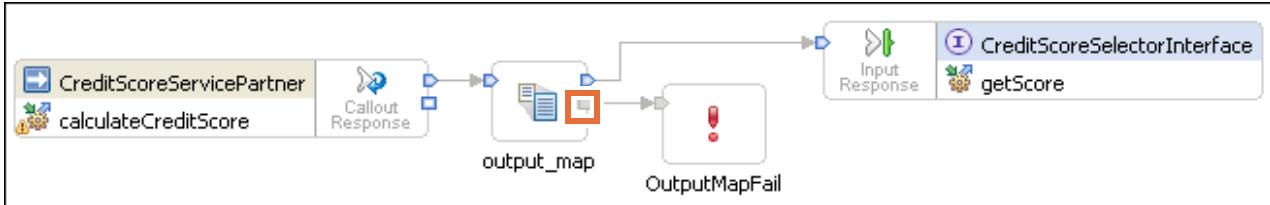
- d. Click the **Local map** transformation.
- 27. In the local map, click the **Auto map input to output** icon (the fifth icon from the right). In the **Auto map** window, accept the default values and click **Finish**. It creates a series of Move transformations from the attributes in the source object to the attributes in the target object.
- 28. At the top of the local map, click the **Up a level** icon (the up arrow) to exit.



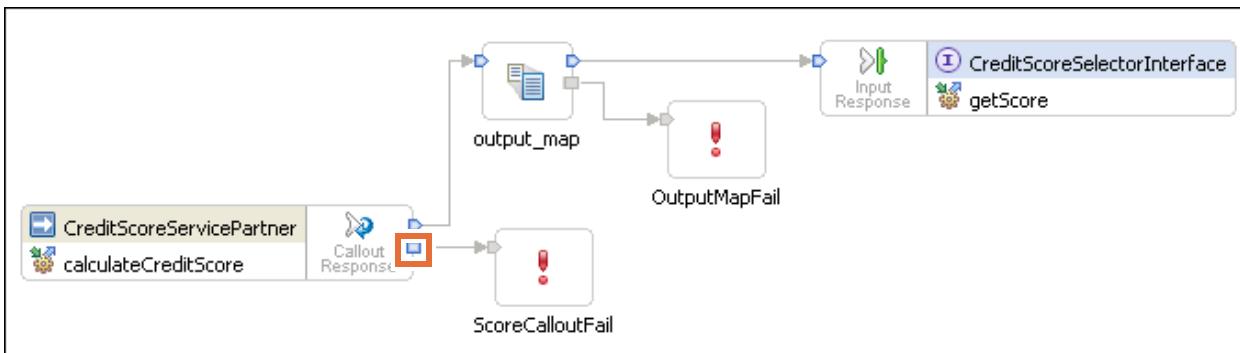
- 29. Save your changes and close the data map editor.
- 30. Add a **Fail** node to the **Response** flow, which throws an exception for a failure in the mapping primitive.

  - a. In the **Palette**, expand the **Error Handling** drawer and click **Fail**.
  - b. Click the flow diagram just below `output_map` to add it to the diagram.
  - c. Change the display name from `Fail1` to: `OutputMapFail`
  - d. Hover the mouse pointer over the **Fail** terminal on the `output_map` primitive.
  - e. Drag the orange, circular **Add wire** handle to the **in** terminal on the **Fail** primitive.

- \_\_\_ f. Save your changes. The Response flow must look like the following screen capture:



- \_\_\_ 31. Add a **Fail** node to the **Response** flow that throws an exception for a failure in the **CalloutResponse** node.
- In the **Palette**, expand the **Error Handling** drawer and click **Fail**.
  - Click the flow diagram just below **CreditScoreServicePartner** to add it to the diagram.
  - Change the display name from `Fail1` to: `ScoreCalloutFail`
  - Hover the mouse pointer over the **Fail** terminal on the **CalloutResponse** node.
  - Drag the orange, circular **Add wire** handle to the **in** terminal on the **Fail** primitive.
  - Save your changes. The Response flow must look like the following screen capture:



- \_\_\_ 32. Close the flow editor.  
\_\_\_ 33. Save your changes to the **CreditScoreMediationService** assembly diagram.



### Important

If you receive the following error in the Problems view, verify that the Transaction qualifier setting for the **CreditScoreMediationService** implementation is set to **Global: CWSKA8018E: The joinTransaction qualifier defined on the interface and the transaction qualifier on the implementation do not match**.

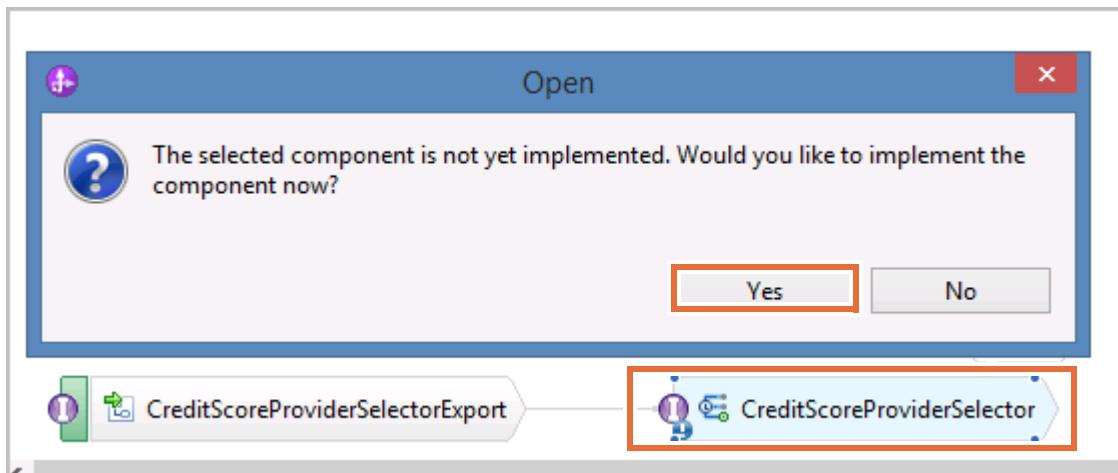
- Switch back to the **CreditScoreMediationService** assembly diagram.
- Click the canvas.
- In the **Properties** view, click **All Qualifiers**.
- Expand **CreditScoreMediationService**.

- Set the Transaction qualifier for Implementation to **Global**.

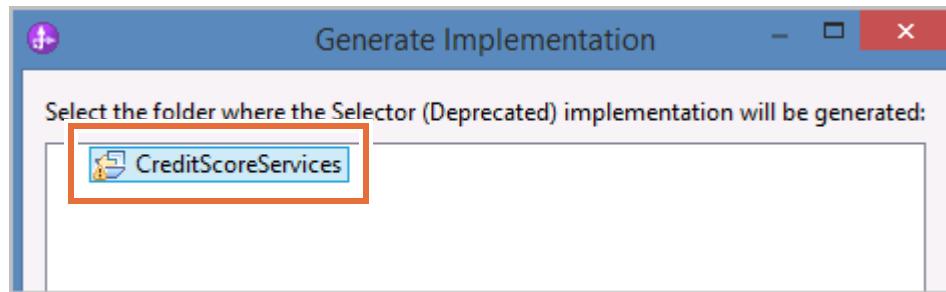
The following table shows the qualifiers that determine the Quality of Service (QoS) for the component.

| Reliability                       |               |              |             |              |
|-----------------------------------|---------------|--------------|-------------|--------------|
| Component                         | Join tran...  | Transacti... | Suspend ... | Asynchron... |
| CreditScoreMediationServiceExport |               |              |             |              |
| CreditScoreMediationService       |               |              |             |              |
| References                        |               |              |             |              |
| Interfaces                        | <True>        |              |             |              |
| Implementation                    | <b>Global</b> |              |             |              |
| CreditScoreServiceImport          | <False>       |              |             |              |

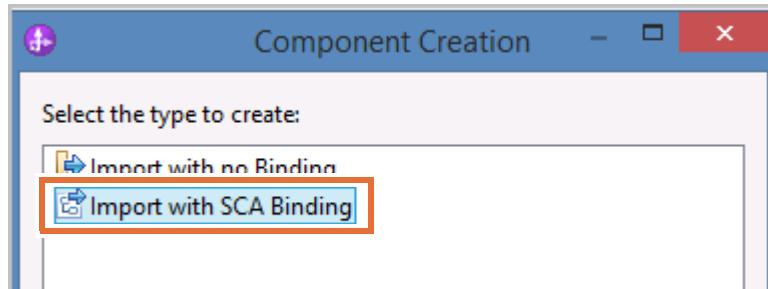
34. Close the **CreditScoreMediationService** assembly editor.
35. Generate the **CreditScoreProviderSelector** component implementation.
  - a. On the **CreditScoreServices** assembly diagram, double-click the **CreditScoreProviderSelector** component to open the selector editor.
  - b. If you receive an **Open** window, click **Yes** to implement the component.



- \_\_\_ c. In the **Generate Implementation** window, leave **CreditScoreServices** selected. If the option has a warning marker (a yellow triangle icon), it is safe to ignore it.

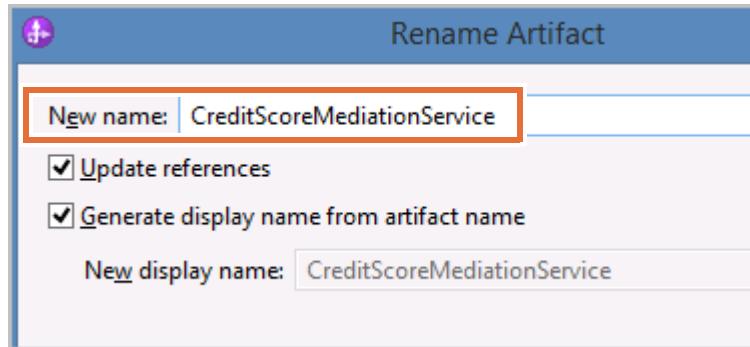


- \_\_\_ d. Click **OK**.
- \_\_\_ e. Close the **Deprecated Selector** note.
- \_\_\_ 36. Add an import component that is named **CreditScoreMediationService** to the **CreditScoreServices** assembly diagram. The selector component calls this import. Imports for the other credit score services are created for you.
- \_\_\_ a. Switch to the **CreditScoreServices - Assembly Diagram** tab.
- \_\_\_ b. In the **Business Integration** view, expand **CreditScoreMediationService > Assembly Diagram**.
- \_\_\_ c. Drag **CreditScoreMediationServiceExport** onto the **CreditScoreServices** assembly diagram.
- \_\_\_ d. In the **Component Creation** window, choose **Import with SCA binding**.



- \_\_\_ e. Click **OK**.
- \_\_\_ f. Save your changes.
- \_\_\_ g. Make sure that the **Import1** component is selected, and switch to the **Description** tab in the **Properties** view.
- \_\_\_ h. Place the cursor in the **Name** field and press **Alt+Shift+R** to refactor it.

- \_\_\_ i. In the **Rename Artifact** window, in the **New Name** field, enter:  
CreditScoreMediationService

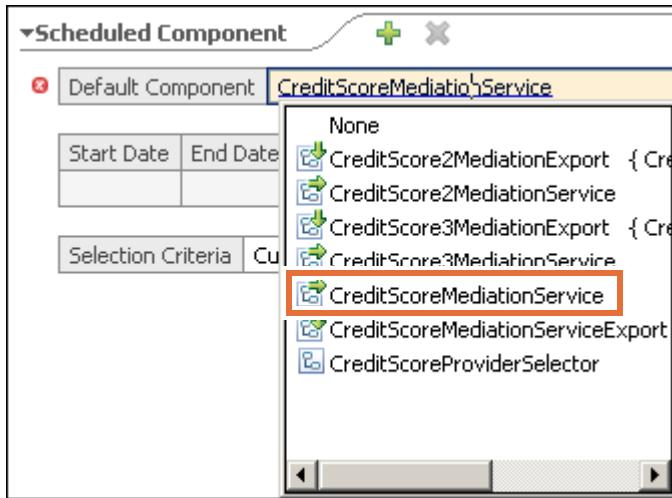


- \_\_\_ j. Accept the remaining default options and click **Refactor**.  
\_\_\_ k. For readability, right-click the **CreditScoreServices** assembly diagram and choose **Layout Contents** from the menu.  
\_\_\_ l. Save your changes. The CreditScoreServices assembly diagram must look like the following screen capture:

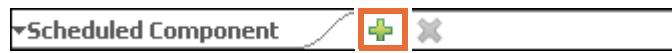


- \_\_\_ 37. Configure the **Scheduled Component** table of the selector. The selector selects the module to invoke based on the date ranges specified in this table. To test this lab, configure this table such that the selector selects the **CreditScoreMediationService** module, and the other modules can be configured to be invoked for the subsequent years. For example, if your current system year is 2017, **CreditScoreMediationService** is invoked between **1/1/17** and **12/31/17**, **CreditScore2MediationService** is invoked between **1/1/18** and **12/31/18**, and **CreditScore3MediationService** is invoked between **1/1/19** and **12/31/19**. The **Default Component** is **CreditScoreMediationService**.
- \_\_\_ a. Switch to the **CreditScoreProviderSelector** tab.  
\_\_\_ b. In the **Default Component** area of the editor, click the **Enter SCA Component** link.

- \_\_\_ c. In the list, choose **CreditScoreMediationService**.



- \_\_\_ d. Click the **Add Date Selection Entry** icon to add a new scheduled component.



- \_\_\_ e. Click the calendar icon in the **Start Date** column and choose **January 1, 2017 at 12:00 AM**. You can also type the values in the field manually. Be sure to leave a space between the comma and the year.
- \_\_\_ f. Click the calendar icon in the **End Date** column and choose **December 31, 2017 at 11:59 PM** (be sure to use PM and not AM). If the default value is incorrect, you must edit the time value manually.
- \_\_\_ g. In the **Component** column, click the **Enter SCA Component** link and choose **CreditScoreMediationService** from the list. The completed entry must look like the following screen capture:

| Start Date           | End Date              | Component                   |
|----------------------|-----------------------|-----------------------------|
| Jan 1, 2017 12:00 AM | Dec 31, 2017 11:59 PM | CreditScoreMediationService |
| Selection Criteria   |                       | Current date                |
|                      |                       |                             |



### Information

Either the Start Date or the End Date can be left blank using the No Date in the date picker. If the Start Date is empty, the component is invoked as soon as the application is deployed. If the End Date is empty, the component is invoked indefinitely.

- \_\_\_ h. Click the **Add Date Selection Entry** icon.
- \_\_\_ i. In the **Start Date** column, click the Calendar icon and choose **January 1, 2018 at 12:00 AM**.

- \_\_ j. In the **End Date** column, click the Calendar icon and choose **December 31, 2018 at 11:59 PM**. If the default value is incorrect, you must edit the time value manually. Be sure to use PM and not AM.
- \_\_ k. In the **Component** column, click the **Enter SCA Component** link and choose **CreditScore2MediationService**. The completed entry must look like the following screen capture:

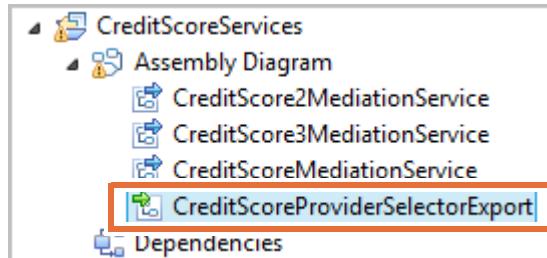
| Start Date           | End Date              | Component                    |
|----------------------|-----------------------|------------------------------|
| Jan 1, 2017 12:00 AM | Dec 31, 2017 11:59 PM | CreditScoreMediationService  |
| Jan 1, 2018 12:00 AM | Dec 31, 2018 11:59 PM | CreditScore2MediationService |
| Selection Criteria   |                       | Current date                 |

- \_\_ l. Click the **Add Date Selection Entry** icon.
- \_\_ m. In the **Start Date** column, click the Calendar icon and choose **January 1, 2019 at 12:00 AM**.
- \_\_ n. In the **End Date** column, click the Calendar icon and choose **December 31, 2019 at 11:59 PM**. If the default value is incorrect, you must edit the time value manually. Be sure to use PM and not AM.
- \_\_ o. In the **Component** column, click the **Enter SCA Component** link and choose **CreditScore3MediationService**. The completed entry must look like the following screen capture:

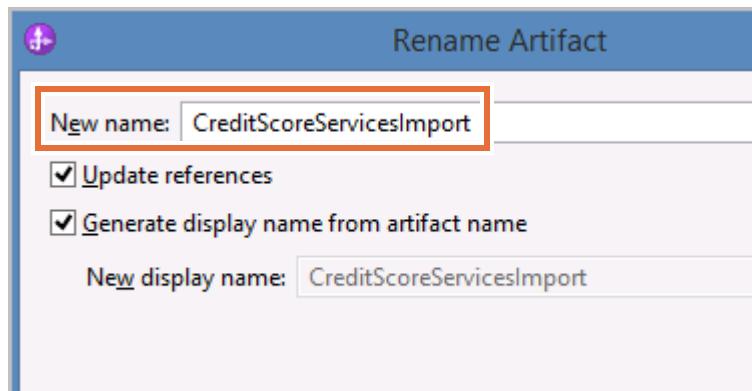
| Start Date           | End Date              | Component                    |
|----------------------|-----------------------|------------------------------|
| Jan 1, 2017 12:00 AM | Dec 31, 2017 11:59 PM | CreditScoreMediationService  |
| Jan 1, 2018 12:00 AM | Dec 31, 2018 11:59 PM | CreditScore2MediationService |
| Jan 1, 2019 12:00 AM | Dec 31, 2019 11:59 PM | CreditScore3MediationService |
| Selection Criteria   |                       | Current date                 |

- \_\_ p. Leave the **Selection Criteria** set to **Current Date** and save your changes.
- \_\_ q. Close the **CreditScoreProviderSelector** tab.
- \_\_ 38. Save and close the **CreditScoreServices - Assembly Diagram** tab.
- \_\_ 39. On the **FoundationModule** assembly diagram, add a **CreditScoreServicesImport** and wire it to the **AccountVerification** business process. You already altered the Credit Check Service invoke activity to call the CreditScoreServices module. The final step in configuring the process is to create the necessary import and wire it to the process component on the assembly diagram. It adds a reference to the process component as well.
  - \_\_ a. In the **Business Integration** view, expand **FoundationModule** and double-click **Assembly Diagram**.

- \_\_ b. In the **Business Integration** view, expand **CreditScoreServices > Assembly Diagram**.
- \_\_ c. Drag **CreditScoreProviderSelectorExport** onto the **FoundationModule** assembly diagram.

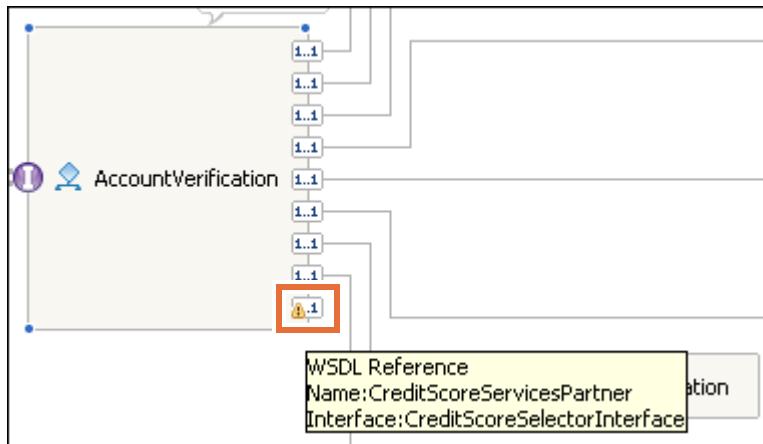


- \_\_ d. In the **Component Creation** window, choose **Import with SCA Binding**.
- \_\_ e. Click **OK**.
- \_\_ f. Save your changes.
- \_\_ g. With **Import1** selected, switch to the **Description** tab in the **Properties** view.
- \_\_ h. Place the cursor in the **Name** field and press **Alt+Shift+R** to refactor it.
- \_\_ i. In the **Rename Artifact** window, in the **New Name** field, enter:  
CreditScoreServicesImport



- \_\_ j. Accept the remaining default options and click **Refactor**.
- \_\_ k. Right-click the **AccountVerification** process component and choose **Synchronize Interfaces and References > from Implementation**.

- \_\_\_ l. When prompted, click **Yes** to synchronize. It adds a reference for the **CreditScoreServicesPartner** to the component.



- \_\_\_ m. Right-click **CreditScoreServicesImport** and choose **Wire to Existing** from the menu.
- \_\_\_ n. Save your changes and verify that the Problems view contains no errors.

## **Part 2: Testing the selector**

Now that a selector component is added to the implementation, the next step is to test it using the integrated test client. To test the selector component:

- \_\_\_ 1. If the server is not started, start it.
  - \_\_\_ a. In the **Servers** view, right-click the **IBM Process Server v8.5.7 at localhost** server.
  - \_\_\_ b. Wait for the server to start before proceeding. The **State** column shows **Started** and the **Server Logs** view shows the message: **Server server1 open for e-business**.

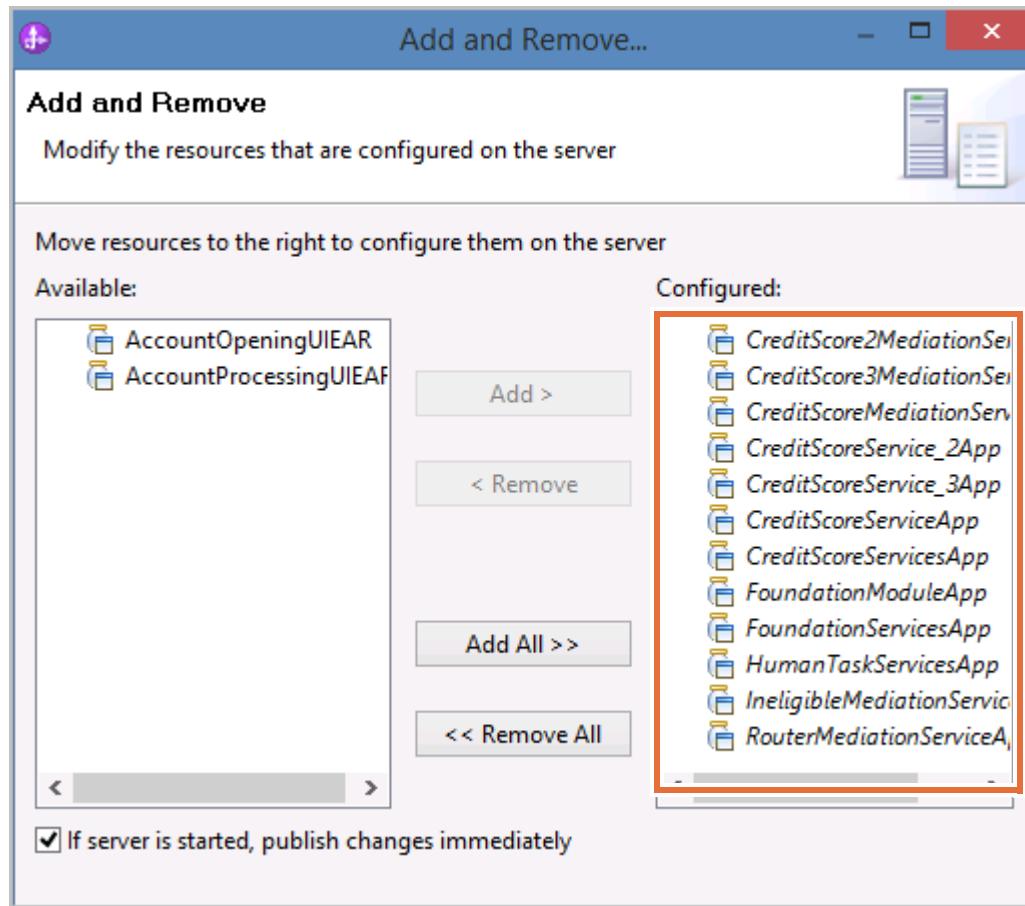


### Note

If the server start status shows for more than 5 minutes and the **Server Logs** view is not getting updated with server start messages, then you need to start it manually. Right-click **IBM Process Server v8.5.7 at localhost** and select **Stop** from the menu to stop successfully (do the Stop operation one more time if the server status does not show as Stopped). Restart the server by right-clicking **IBM Process Server v8.5.7 at localhost** and selecting **Start**.

- 
- \_\_\_ 2. Deploy the modules to the server.
    - \_\_\_ a. In the **Servers** view, right-click **IBM Process Server v8.5.7 at localhost**.
    - \_\_\_ b. Select **Add and Remove** from the menu.

- c. Use **Add** to add all of the projects to the Configured list except for AccountOpeningUIEAR and AccountProcessingUIEAR.



- d. Click **Finish**. The projects are deployed to the server. Wait for the deployment process to complete. It might take several minutes. The State column for each of the applications changes to **Started** in the **Servers** view. Additionally, in the **Server Logs** view, you see several messages that are shown to indicate that the deployed applications started successfully.
- 3. Test the application by using **companyName IBM**. Depending on when this test is run, the resulting credit score is either 9, 10, or 11. Recall from the selector component criteria that three possible mediation services can be invoked. **CreditScoreMediationService** is invoked between **1/1/17** and **12/31/17** and returns a **credit score** of 11. **CreditScore2MediationService** is invoked between **1/1/18** and **12/31/18** and returns a **credit score** of 10. **CreditScore3MediationService** is invoked between **1/1/19** and **12/31/19** and returns a **credit score** of 9. The result varies depending on the year in which the service is invoked. Any year other than 2017, 2018, or 2019 results in invoking the **Default Component**, which is **CreditScoreMediationService**, in which case the credit score is always 11.
  - a. In the **Business Integration** view, right-click **FoundationModule** and choose **Test > Test Modules** from the menu.
  - b. On the integrated test client tab, go to the **Initial Request Parameters** section.

- \_\_\_ c. In the **Value** column, enter **IBM** for the **companyName** field.

Initial request parameters:

Value editor  XML editor

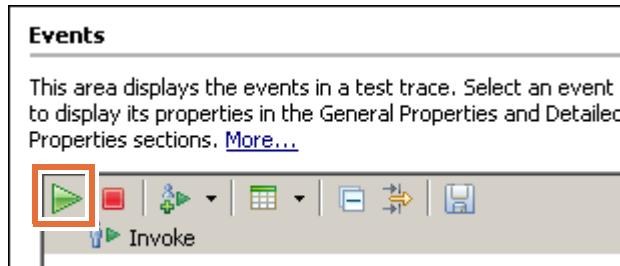
| Name                | Type    | Value      |
|---------------------|---------|------------|
| accountNumber       | string  | [ab]       |
| applicationDate     | string  | [ab]       |
| applicationDecision | boolean | [ab] false |
| comments            | string  | [ab]       |
| companyName         | string  | [ab] IBM   |
| contactFirstName    | string  | [ab]       |



### Note

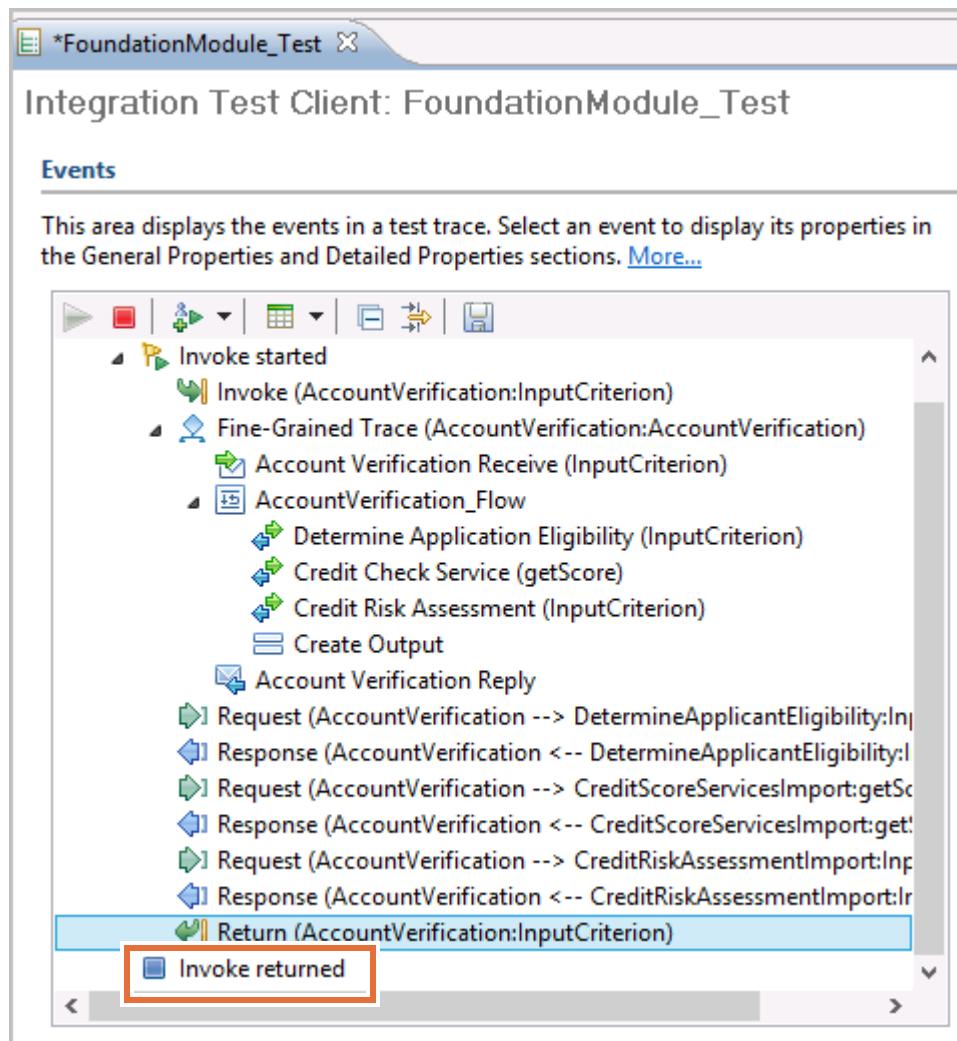
You can maximize the input table to simplify data entry.

- \_\_\_ d. Click **Continue** on the Events toolbar.



- \_\_\_ e. If you are prompted for the **Deployment Location**, leave **IBM Process Server v8.5.7 at localhost** as selected and click **Finish**.
- \_\_\_ f. At the **User Login** window, accept the default values for **User ID** and **Password** and click **OK**.

- g. Because the test case for IBM results in automatic approval, the test must run to completion, which is indicated in the blue, square stop node.



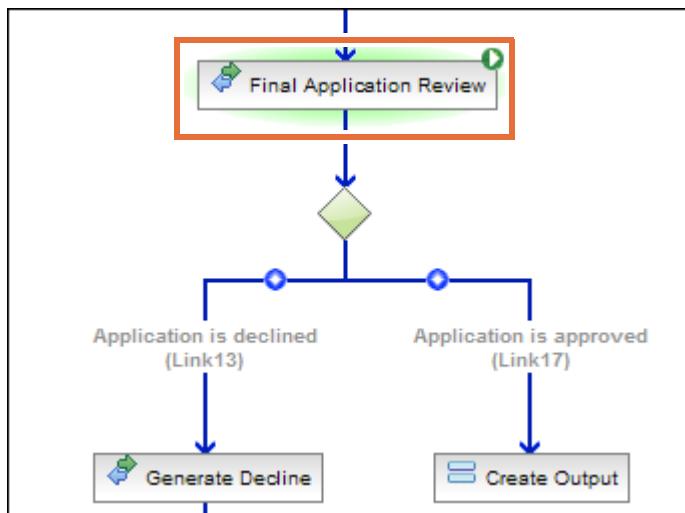
- h. In the **Events** window, select the **Response (AccountVerification <-- CreditScoreServicesImport:getScore)** event.
- i. Depending on the year, the **CreditScoreService** returns a credit score of either 9, 10, or 11.
- j. Since you set the Selector to invoke services that are based on the date ranges, it invokes the **CreditScoreService**, **CreditScoreService2**, or **CreditScoreService3**, which returns a credit score of 9, 10, or 11.
- 4. Leave the server still running with the applications deployed. You use them in the next section. Close all open tabs. Do not save FoundationModule\_test.

# Exploring process dynamicity

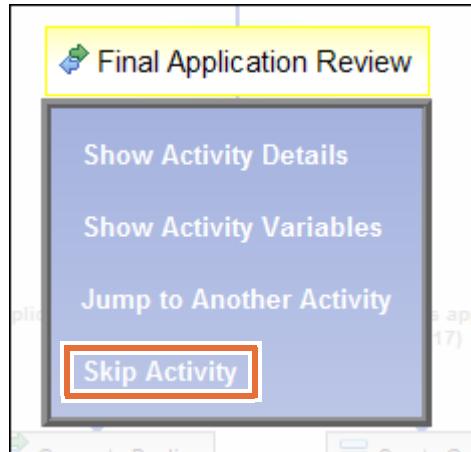
In this portion of the exercise, you use the Business Process Choreographer Explorer client to skip an activity in a running process instance.

To skip an activity:

- \_\_\_ 1. Run a test of the **FoundationModule** by using **companyName ACME**.
  - \_\_\_ a. In the **Business Integration** view, right-click **FoundationModule** and choose **Test > Test Modules** from the menu.
  - \_\_\_ b. In the **Initial request parameters** section of the integrated test client, enter **ACME** for the **companyName**.
  - \_\_\_ c. Click **Continue** on the Events toolbar.
  - \_\_\_ d. At the **User Login** window, accept the default values for **user ID** and **Password**, and click **OK**.
- \_\_\_ 2. Use the **Business Process Choreographer Explorer** client to skip the Final Application Review task.
  - \_\_\_ a. In the **Servers** view, right-click **IBM Process Server v8.5.7 at localhost** and choose **Launch > Business Process Choreographer Explorer** from the menu.
  - \_\_\_ b. If you receive a security alert, click **Yes** to continue.
  - \_\_\_ c. Use **User Name** **admin** and **Password** **web1sphere** and click **Login**.
  - \_\_\_ d. In the navigator pane, click **Process Instances > Started by me**. You see one active process instance.
  - \_\_\_ e. Select a check box to the left of the process instance and click **View Process State**.
  - \_\_\_ f. The process must show that Final Application Review is the current activity. You need to scroll down on Process State View to see the Final Application Review activity.



- \_\_ g. Click **Final Application Review** and choose **Skip Activity** from the menu.



- \_\_ h. You receive a status message that indicates that the skip request was successful. Close the message window by clicking the X icon.



#### Note

If you receive a warning or an error that the requested object does not exist, you can safely ignore it as the process was completed when you skipped the activity. Click **Exit** or close the message window by clicking the **X** icon.

- 
- \_\_ i. Click **Process Instances > Started by me**. You no longer see any active process instance.  
\_\_ j. Log out of the Business Process Choreographer Explorer client.

- k. Switch to the integrated test client tab. See the details in the Event tab and note the skipped status next to Final Application Review event. You also see a blue, square stop node that indicates the test is complete.

**Integration Test Client: FoundationModule\_Test**

**Events**

This area displays the events in a test trace. Select an event to display its properties in the General Properties and Detailed Properties sections. [More...](#)

```

graph TD
 Start[Start] --> Inv1[Invoke (AccountVerification:InputCriterion)]
 Inv1 --> InvStarted[Invoke started]
 InvStarted --> Inv2[Invoke (AccountVerification:InputCriterion)]
 Inv2 --> Trace[Fine-Grained Trace (AccountVerification:AccountVerification)]
 Trace --> AV[Account Verification Receive (InputCriterion)]
 Trace --> Flow[AccountVerification_Flow]
 Flow --> DA[Determine Application Eligibility (InputCriterion)]
 Flow --> CCS[Credit Check Service (getScore)]
 Flow --> CRA[Credit Risk Assessment (InputCriterion)]
 Flow --> FAR[Final Application Review (InputCriterion) [Skipped]]
 FAR --> CreateOutput[Create Output]
 CreateOutput --> AVReply[Account Verification Reply]
 AVReply --> DAReq[Request (AccountVerification --> DetermineApplicantEligibility:InputCriterion)]
 DAReq --> DARes[Response (AccountVerification <-- DetermineApplicantEligibility:InputCriterion)]
 DARes --> CCSReq[Request (AccountVerification --> CreditScoreServicesImport:getScore)]
 CCSReq --> CCSRes[Response (AccountVerification <-- CreditScoreServicesImport:getScore)]
 CCSRes --> CRAReq[Request (AccountVerification --> CreditRiskAssessmentImport:InputCriterion)]
 CRAReq --> CRARes[Response (AccountVerification <-- CreditRiskAssessmentImport:InputCriterion)]
 CRARes --> FARReq[Request (AccountVerification --> FinalApplicationReview:InputCriterion) [Running]]
 FARReq --> FARReturn[Return (AccountVerification:InputCriterion)]
 FARReturn --> InvReturned[Invoke returned]

```

Events Configurations

- l. Close the test client tab and click **No** when prompted to save the test trace.
- 3. Remove the applications.
- a. In the **Servers** view, right-click **IBM Process Server v8.5.7 at localhost** and choose **Add and Remove** from the menu.
  - b. Click **Remove All** and click **Finish**.
- 4. Close the IBM Integration Designer. Click **File > Exit**, or click **X** in the upper-right corner.

## End of exercise

# Exercise 9. Working with static relationships

## Estimated time

01:00

## Overview

In this exercise, you implement a static relationship in an existing SCA module.

## Objectives

After completing this exercise, you should be able to:

- Implement a static relationship to map corresponding values between business object fields
- Import static relationship data from a comma-separated value file
- Test a relationship in the IBM Integration Designer integrated test environment

## Introduction

When you integrate multiple applications, it is not unusual to represent commonly used data elements in multiple formats. For example, one system might represent state or province names by using two-letter abbreviations, while other systems might represent the same values by using numeric representations. Rather than writing application logic to do these translations, you can create a relationship in IBM Integration Designer that can be invoked from a data map.

You modify the WebSphere MQ binding exercise to include a static relationship. In this case, it is determined that the customer service application, which receives the “declined application” messages sent to it through WebSphere MQ, uses a numeric code to represent the Country field in the credit application. However, the sending application uses a character string to represent the Country field. You create the static relationship, integrate it into the existing application, and then verify that the relationship works as expected by testing it.

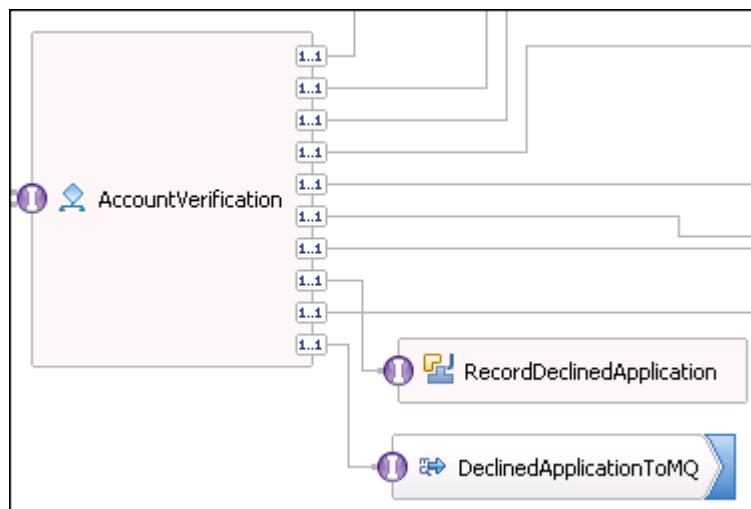
## Requirements

To complete the exercises for this course, you need a lab environment, as follows: this environment includes the exercise support files, Mozilla Firefox, IBM Integration Designer V8.5.7, IBM DB2 V10.1, RFHUTIL, WebSphere MQ V8.0, and IBM Process Server V8.5.7 test environment.

## Part 1: Create the lookup relationship

The workspace already contains a module that you use to implement the relationship.

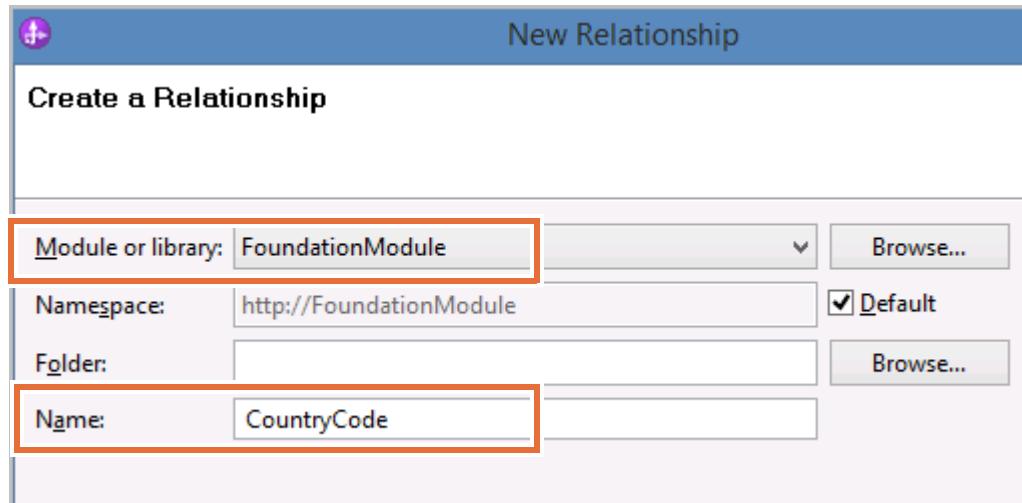
- 1. Open the Exercise 9 workspace.
  - a. On your desktop, open the folder that is labeled **Exercise Shortcuts**.
  - b. Double-click the shortcut that is labeled **Exercise 9**. IBM Integration Designer starts.
  - c. Wait for the workspace build to complete; the build might take a few minutes.
  - d. Close the **Getting Started** tab.
- 2. Examine the assembly diagram for the `FoundationModule`.
  - a. In the **Project** section of the **Business Integration** view, expand `FoundationModule`.
  - b. Double-click **Assembly Diagram**. The assembly diagram opens in the editor pane.



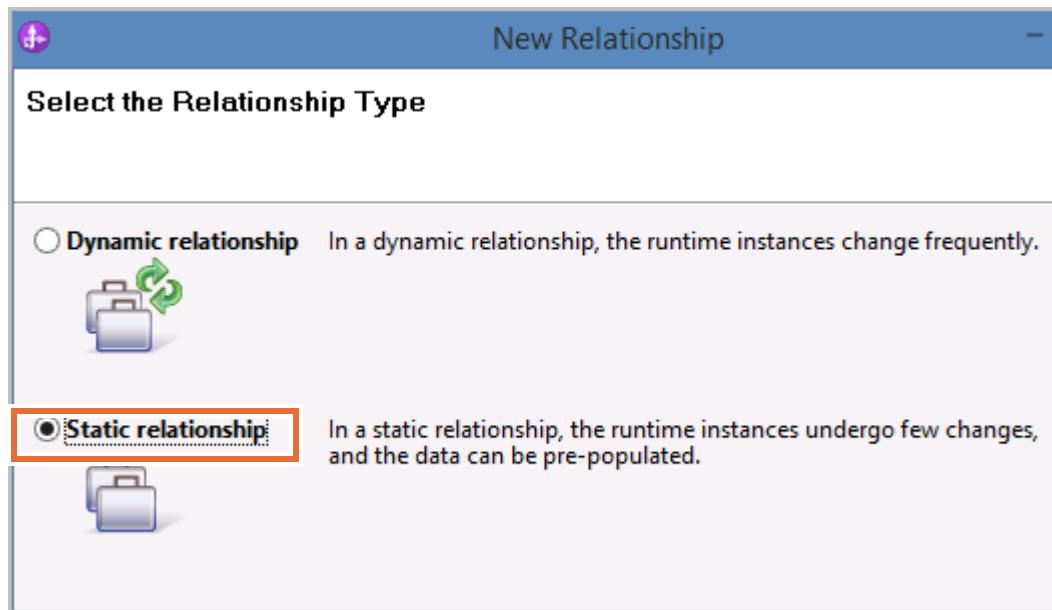
Notice the `DeclinedApplicationToMQ` component, which you added in Exercise 2. Recall that this component writes information about declined applications to a WebSphere MQ queue.

- 3. Create the relationship.
  - a. From the menu options, choose **File > New > Relationship**. The **New Relationship** window is opened.
  - b. In the **Module or Library** field, choose `FoundationModule` from the list.

- \_\_ c. In the **Name** field, enter: CountryCode

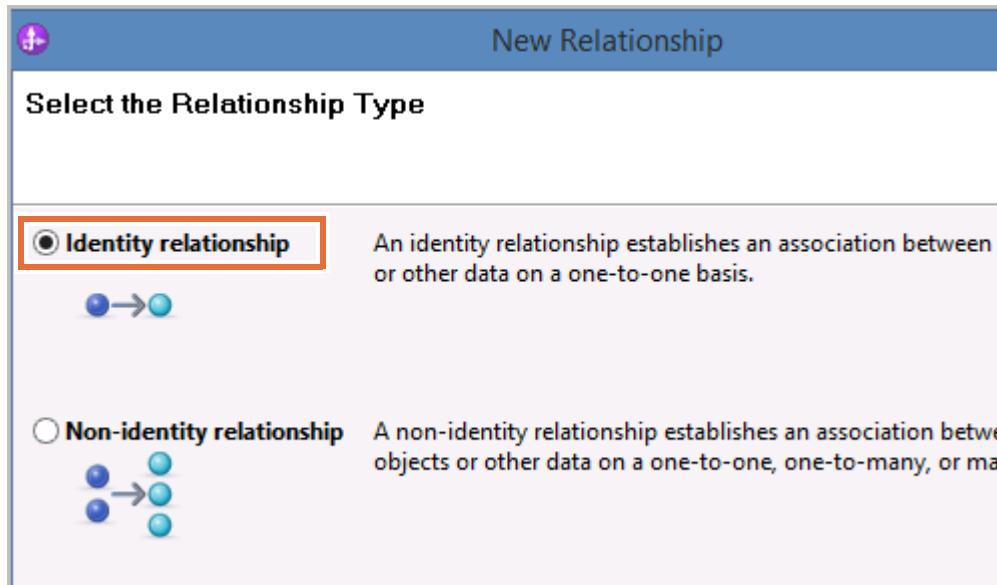


- \_\_ d. Click **Next**.  
\_\_ e. For the relationship type, select **Static relationship**.



- \_\_ f. Click **Next**.

- \_\_ g. For the relationship type, select **Identity relationship**.

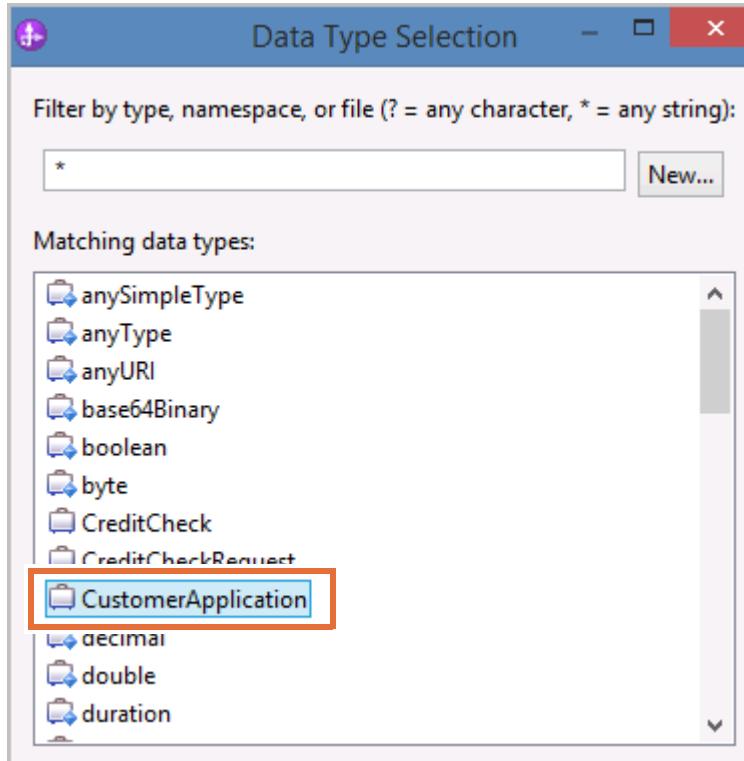


- \_\_ h. Click **Finish**. The relationship is created, and the relationship editor opens.  
\_\_ i. Click **Add role** (the first icon to the right of the Roles section header).



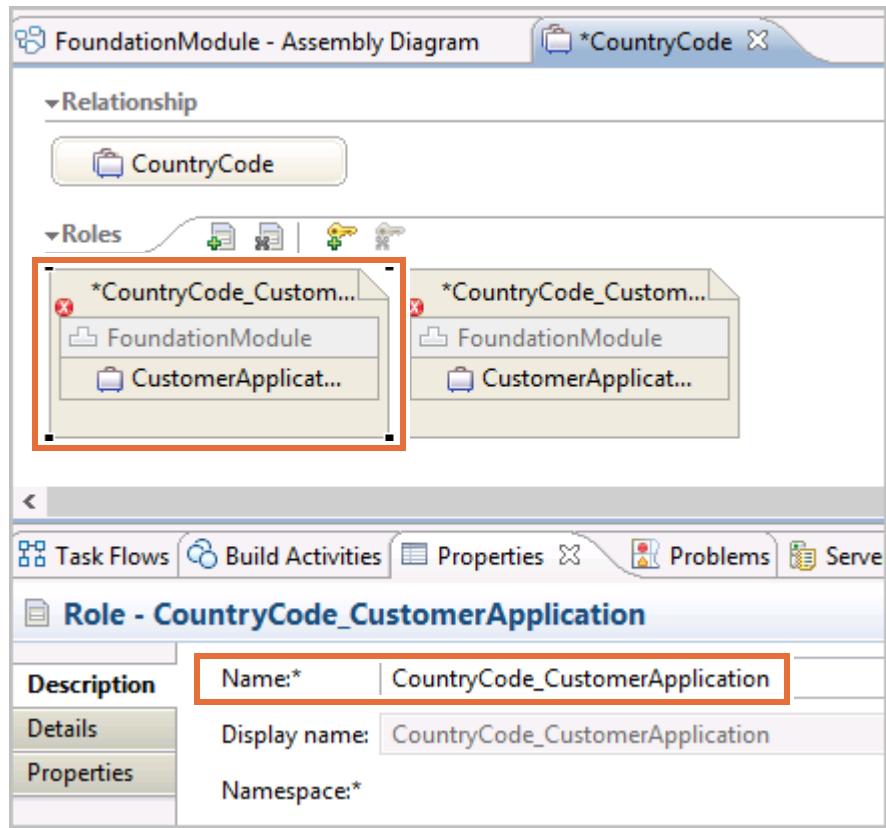
The **Data Type Selection** window opens.

- \_\_ j. Click **CustomerApplication**.



- \_\_ k. Click **OK**. The CustomerApplication role is shown in the editor.
- \_\_ l. Again, click **Add Role**, and the **Data Type Selection** window opens.
- \_\_ m. Again, click **CustomerApplication**, and click **OK**. A second CustomerApplication role opens in the editor. You defined the two roles that participate in the relationship. Now you define the keys: the fields in each business object that are used for the data conversion. If you see errors in the Problems view, it is safe to ignore them.
- \_\_ n. Click the first **CountryCode\_CustomerApplication** role.

- o. Switch to the **Properties** view. Notice in the **Properties** that this role participant is labeled **CountryCode\_CustomerApplication**.



- p. Click **Add key attribute** (the third icon to the right of the **Roles** section header).



The Select Key Attributes window opens.

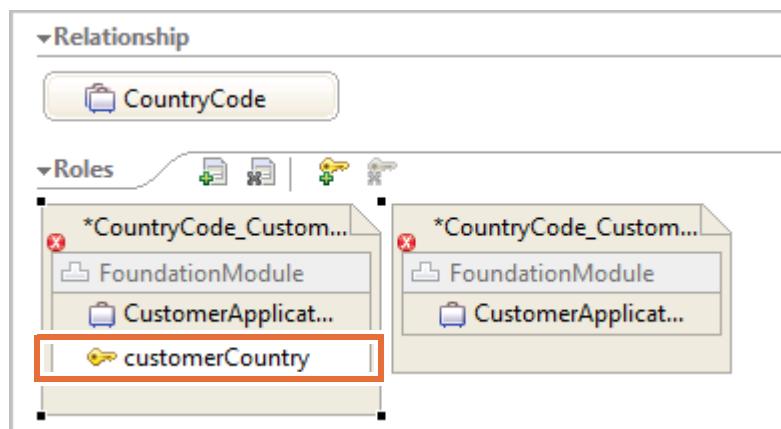
- \_\_ q. Select **customerCountry** from the list.

Select Key Attributes

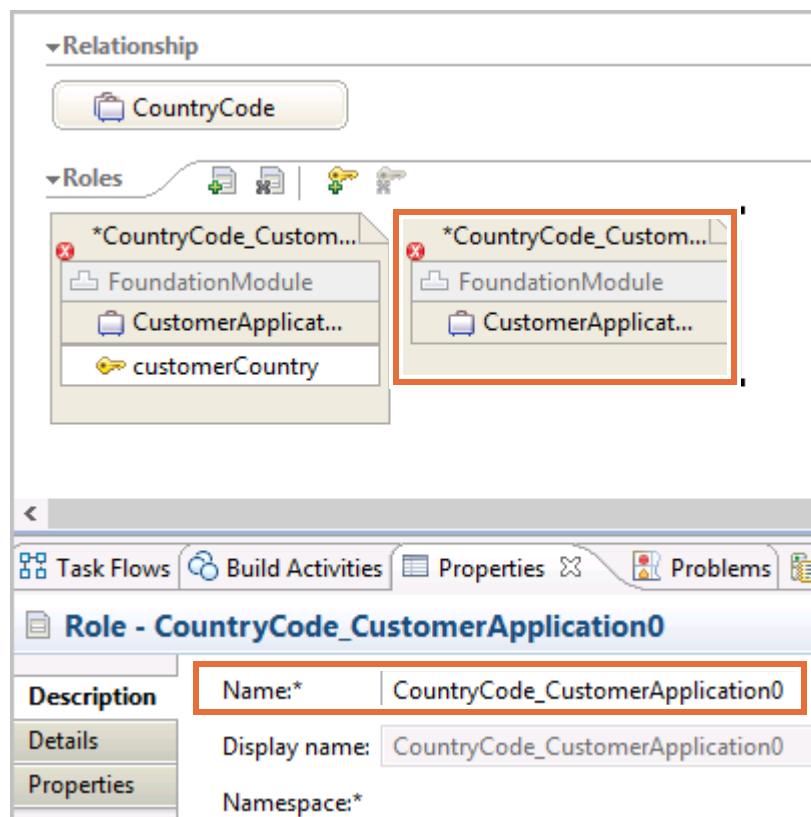
Select key attributes for the combined key of the role.

| Name                 | Type    | Index | Kind        |
|----------------------|---------|-------|-------------|
| contactPhoneNumber   | string  |       | xsd:element |
| creditRating         | string  |       | xsd:element |
| creditReportNeeded   | boolean |       | xsd:element |
| creditRisk           | string  |       | xsd:element |
| creditScore          | int     |       | xsd:element |
| customerCity         | string  |       | xsd:element |
| customerCountry      | string  |       | xsd:element |
| eligibleApplication  | boolean |       | xsd:element |
| ineligibleReason     | string  |       | xsd:element |
| pricingCode          | string  |       | xsd:element |
| pricingScore         | string  |       | xsd:element |
| productName          | string  |       | xsd:element |
| requestAccountAmount | int     |       | xsd:element |

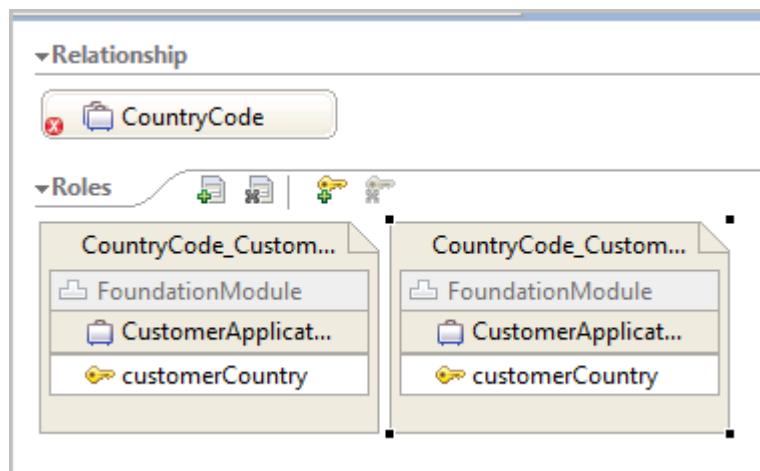
- \_\_ r. Click **OK**. The **customerCountry** key attribute is added in the **CountryCode\_CustomerApplication** role.



- s. Similarly, click the second **CountryCode** role (which is labeled **CountryCode\_CustomerApplication0** in the **Properties** view).



- t. Click **Add key attribute**. The **Select Key Attributes** window opens.  
— u. Select **customerCountry** from the list and then click **OK**.



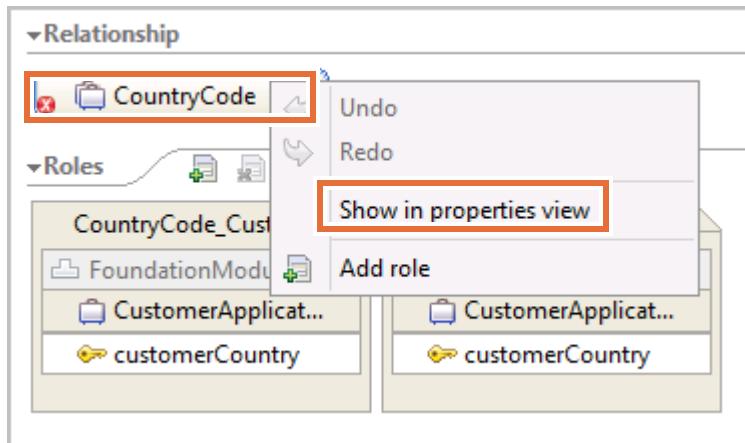
You now defined both roles, and the keys that participate in the relationship.

- v. Save the workspace (**Ctrl+S**). Continue to ignore any errors in the Problems view.  
— w. Import the data that is used for the relationship.

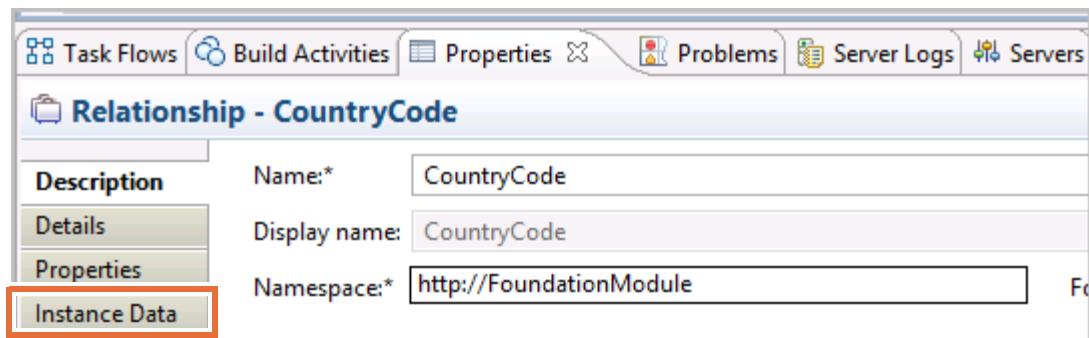
For small amounts of data, the values that are used in a relationship can be entered manually into the instance data table. For large volumes of data, with IBM Integration Designer, you can import the data from a .csv (comma-separated values) file.

For this exercise, you import from a file that contains ISO standard country code values in numeric and alphanumeric formats (ISO standard 3166 defines the two-character and three-character alphanumeric and three-digit numeric codes for countries).

- \_\_\_ x. Right-click the **CountryCode** relationship icon, and select **Show in properties view** from the menu.



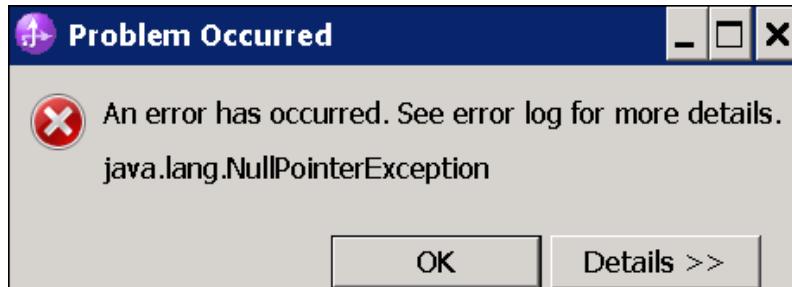
- \_\_\_ y. In the **Properties** view, select the **Instance Data** tab.



The instance data window opens.

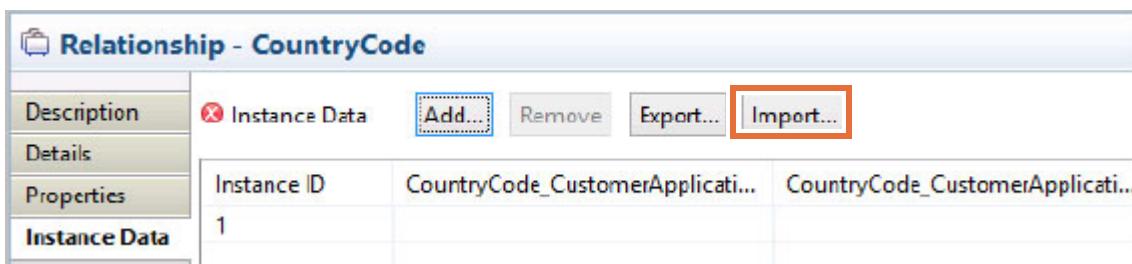
**Note**

If you see a `java.lang.NullPointerException` as shown, you can safely ignore it now as this exception will be resolved after doing subsequent steps. Click **OK** to ignore.



If Instance Data is not visible, click one of the roles and then CountryCode in the Business Integration view to see Instance Data in the Properties view.

z. Click **Import**.



The Import Relationship Instance Data window opens.

**Note**

You can also initiate the import process by selecting **File > Import > Business Integration > Relationship Instance Data**.

- aa. Click **Browse**, go to `C:\labfiles\Support Files\EX9\CountryCodes.csv`, and click **Open**.
- ab. Click **Next**. The **Select the Roles and Key Attributes** window opens.

Here you associate (or “bind”) the data in the file with the specific roles. Notice that each row of data in the file contains four attributes: country name, ISO two-character alphanumeric representation, ISO three-character alphanumeric representation, and ISO numeric representation. Because the relationship you defined has only two roles, you bind only two of the data columns to the relationship: the country name and the ISO numeric representation.

- ac. In column **1** of the table, click in the highlighted cell that is labeled **(Not bound)**. An arrow appears.

- \_\_ ad. Click the arrow and select **CountryCode\_CustomerApplication/customerCountry** (the first entry) from the list.

| 1                                                                                    | 2                                | 3           |
|--------------------------------------------------------------------------------------|----------------------------------|-------------|
| <input type="checkbox"/> (Not bound)                                                 | <input type="button" value="▼"/> | (Not bound) |
| <input checked="" type="checkbox"/> (Not bound)                                      |                                  | ISO 3       |
| <input checked="" type="checkbox"/>  CountryCode_CustomerApplication/customerCountry |                                  | AFG         |
| <input checked="" type="checkbox"/> ALAND ISLANDS                                    | AX                               | ALA         |
| <input checked="" type="checkbox"/> ALBANIA                                          | AL                               | ALB         |
| <input checked="" type="checkbox"/> ALGERIA                                          | DZ                               | DZA         |
| <input checked="" type="checkbox"/> AMERICAN SAMOA                                   | AS                               | ASM         |
| <input checked="" type="checkbox"/> ANDORRA                                          | AD                               | AND         |
| <input type="checkbox"/> ANGOLA                                                      | AO                               | AGO         |

- \_\_ ae. In column 4 of the table, click in the highlighted cell that is labeled **(Not bound)**. An arrow is displayed.
- \_\_ af. Click the arrow and select **CountryCode\_CustomerApplication0/customerCountry** from the list. Notice that this entry is the only available entry, since a role can be bound to only one column of data.

|   |                                                  |
|---|--------------------------------------------------|
| 4 | (Not bound)                                      |
|   | <input type="button" value="▼"/>                 |
|   | CountryCode_CustomerApplication0/customerCountry |
|   | 248                                              |
|   | 8                                                |
|   | 12                                               |
|   | 16                                               |
|   | 20                                               |
|   | 24                                               |

- \_\_ ag. Scroll to the left. Notice that the second row of the table contains row headers (**Country Name, ISO 2, ISO 3, ISO NUMERIC**). As you do not want to import these row headers into the table, *clear* the box beside that row.

|   |                                                       |
|---|-------------------------------------------------------|
| 1 | <input type="checkbox"/> CountryCode_CustomerAppli... |
|   | <input checked="" type="checkbox"/> Country Name      |
|   | <input checked="" type="checkbox"/> AFGHANISTAN       |
|   | <input checked="" type="checkbox"/> ALAND ISLANDS     |

- \_\_ ah. Click **Finish**.  
\_\_ ai. If a message indicates that the relationship instance data exists, click **Overwrite**.

The data is imported. Because you did not bind the **ISO 2** or **ISO 3** columns, they are not imported. The imported data is visible in the instance data table.

| Instance ID | CountryCode_CustomerApplication/customer... | CountryCode_CustomerApplication0/custo.. |
|-------------|---------------------------------------------|------------------------------------------|
| 1           | AFGHANISTAN                                 | 4                                        |
| 2           | ALAND ISLANDS                               | 248                                      |
| 3           | ALBANIA                                     | 8                                        |

- \_\_ 4. Save the workspace (**Ctrl+S**). The Problems view contains no errors.  
\_\_ 5. Close the relationship editor tab.

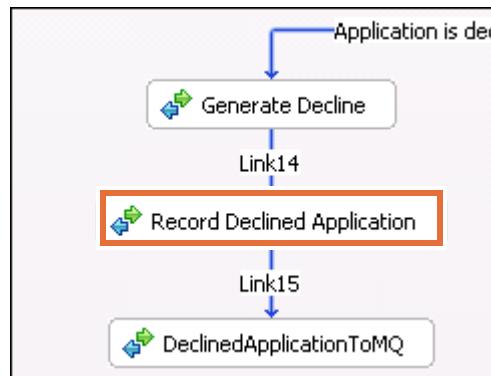
Now that you defined the relationship, you create a data map that invokes it.

## Part 2: Creating a data map to invoke the relationship

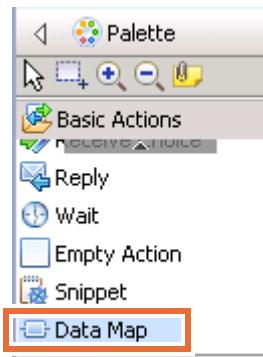
Since the messages that you send to the Customer Service application (through WebSphere MQ) must contain the numeric country code rather than the country name, that translation must occur before the DeclinedApplicationToMQ invoke activity. To do the translation, you add a data map activity that invokes the static relationship you created.

- \_\_ 1. Open the **AccountVerification** process in the BPEL editor.
  - \_\_ a. In the **Business Integration** view, expand **FoundationModule > Integration Logic > BPEL Processes > AccountVerification**.
  - \_\_ b. Double-click the **AccountVerification** component to open it in the BPEL editor.

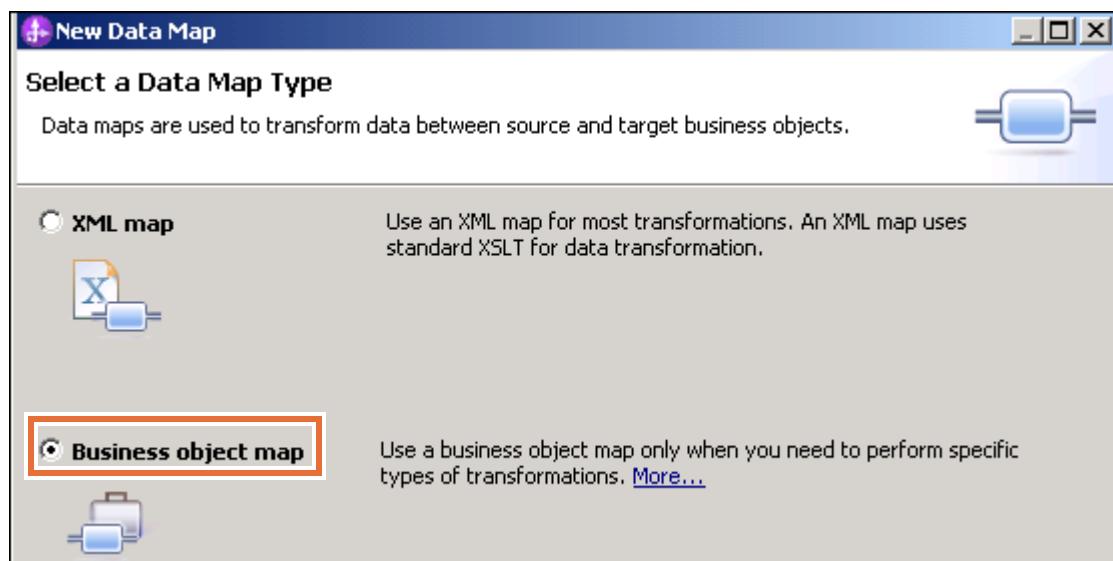
- 2. Scroll to the area where the **Record Declined Application** activity is listed.



- 3. From the **Palette**, in the **Basic Actions** drawer, click **Data Map**.



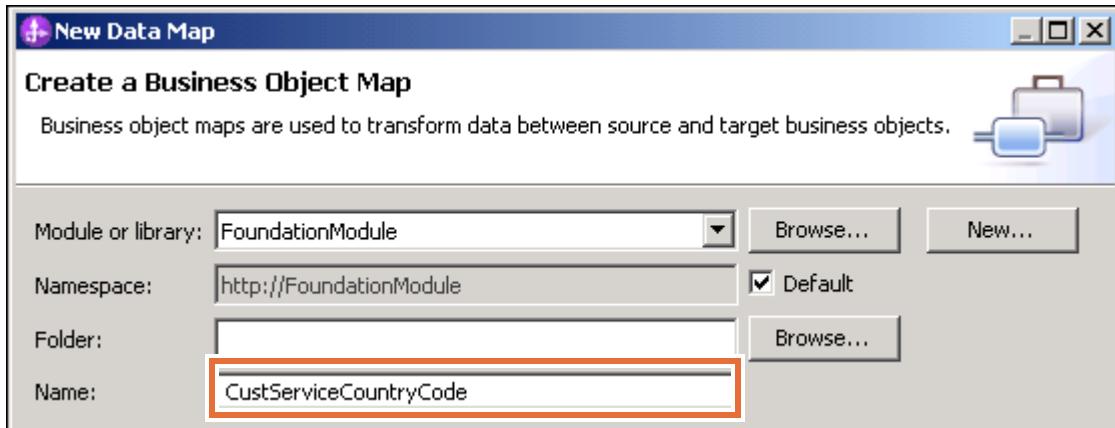
- 4. Position the cursor to the right of the **Record Declined Application** activity, and click to place the data map activity on the diagram. The New Data Map wizard opens.  
 — 5. Configure the data map.  
 — a. At the **Select a Data Map Type** window, select **Business object map**.



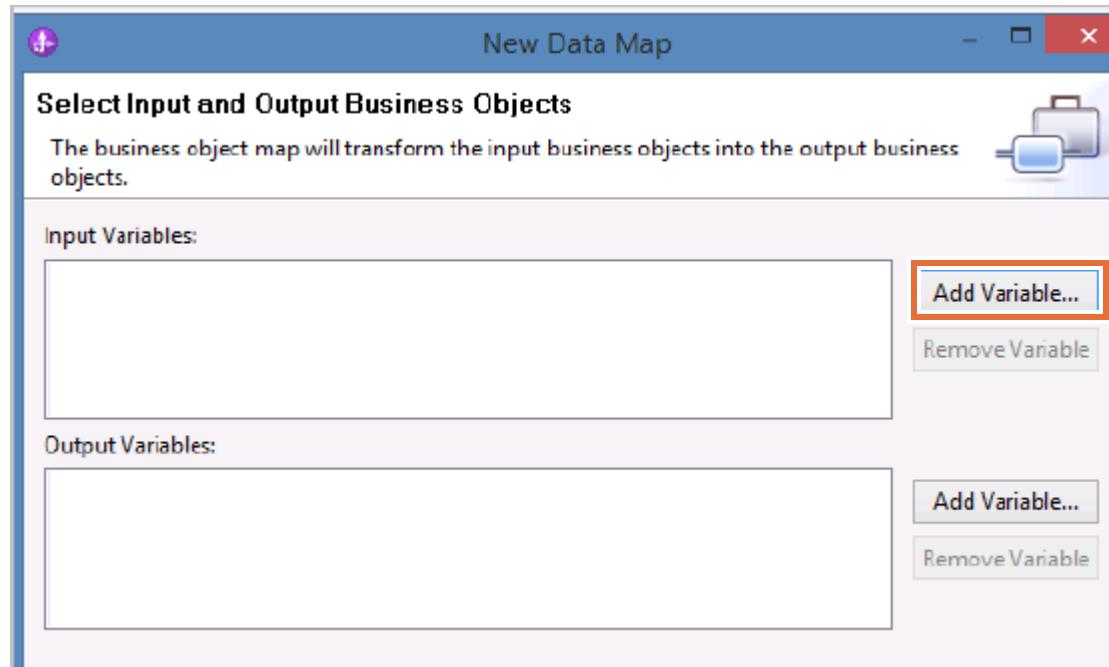
**Note**

You must use a business object map, not an XML map, when you are using a relationship transformation.

- \_\_\_ b. Click **Next**.
- \_\_\_ c. In the **Name** field, enter: *CustServiceCountryCode*



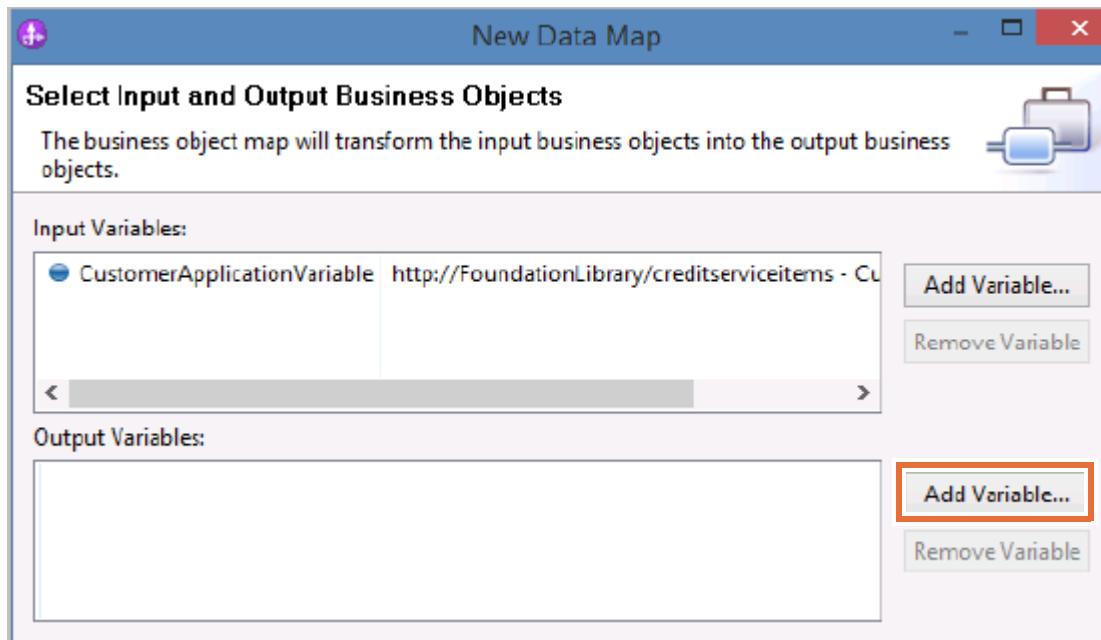
- \_\_\_ d. Click **Next**.
- \_\_\_ e. In the **Select Input and Output Business Objects** window, you define the input and output business objects that are used in the transform. For the **Input Variables**, click **Add Variable** to the right of the field.



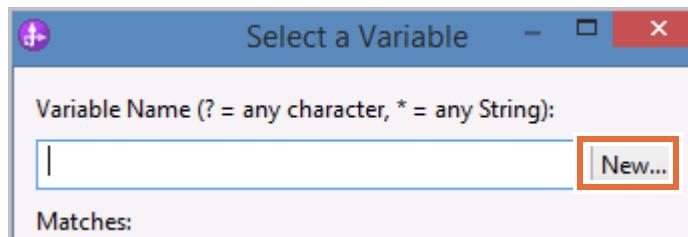
The “Select a variable” window box opens.

- \_\_\_ f. Select **CustomerApplicationVariable** and click **OK**.

- \_\_ g. For the **Output Variables**, click **Add Variable** to the right of the field.

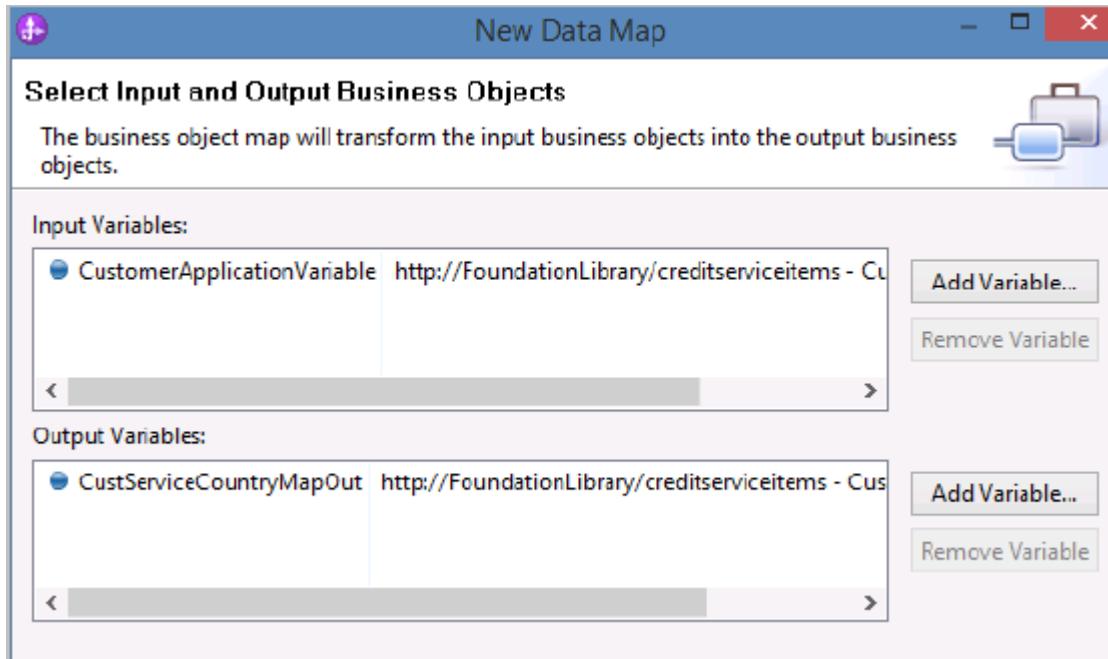


- \_\_ h. To the right of the **Variable Name** field, click **New** to create an output variable to hold the result of the map transformation.



- \_\_ i. In the **Name** field, enter: `CustServiceCountryMapOut`
- \_\_ j. In the **Matching data types** field, click **CustomerApplication** to assign the data type to the new variable.
- \_\_ k. Click **OK**. The new variable is reflected in the **Select a Variable** window.

- \_\_ l. Click **OK**. The input and output variables are listed in the New Data Map window.



- \_\_ m. Click **Finish**. The mapping editor opens with the two business objects shown. The business object on the left represents the “source” of the map operation, and the one on the right represents the “target” of the map operation.
- \_\_ n. Since you want all corresponding fields in the source business object to map to the target business object (other than the country code), click **Map similar fields** (the sixth icon in the **Transformations** section).



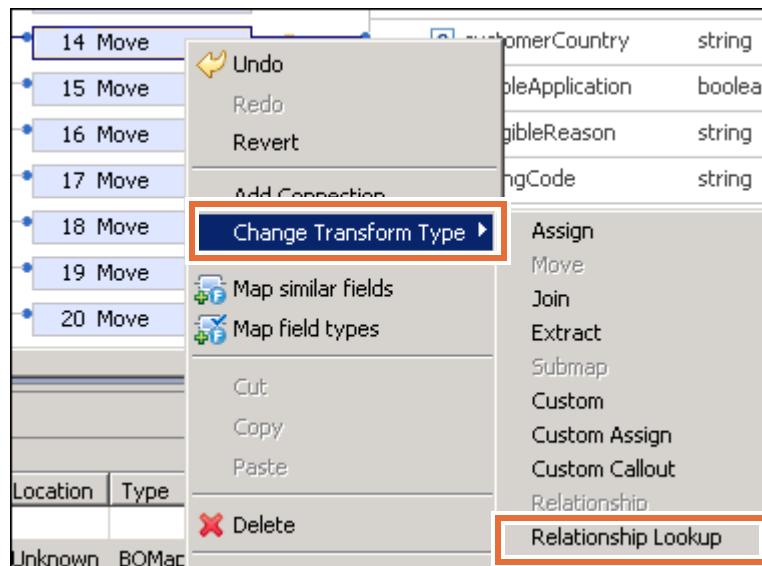
- \_\_ o. A window opens to explain how you can control the operation of the **Map similar fields** transformation. Click **OK** to close it. The fields between the two business objects are now mapped by using **move** transformations.

Now change the country code mapping to use the lookup relationship you defined earlier.

- \_\_ p. Locate the **Move** operation between the two **customerCountry** fields and click to select it. It is transformation 14.



- \_\_\_ q. Right-click the **move** transform, and choose **Change Transform Type > Relationship Lookup** from the menu.



The transformation type changes to Relationship Lookup.

- \_\_\_ r. In the **Properties** view, select the **Details** tab. Here you identify the relationship that you defined earlier and the business objects that participate in the relationship.
- \_\_\_ s. In the **Relationship lookup definition** field, select **CountryCode** from the list.
- \_\_\_ t. In the **Input role name** field, select **CountryCode\_CustomerApplication** from the list.
- \_\_\_ u. In the **Output role name** field, select **CountryCode\_CustomerApplication0** from the list.

| Transform - 14        |                                                                                                                                                                      |
|-----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Description           | Relationship lookup definition: * <input type="text" value="CountryCode"/>                                                                                           |
| Details               | Input role name: * <input type="text" value="CountryCode_CustomerApplication"/><br>Output role name: * <input type="text" value="CountryCode_CustomerApplication0"/> |
| Sources/Targets       |                                                                                                                                                                      |
| Event Monitor         |                                                                                                                                                                      |
| Global Event Settings | Roles for the selected relationship:                                                                                                                                 |

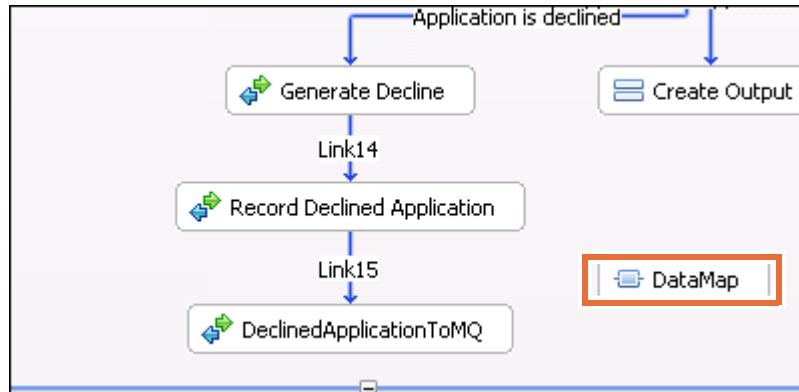
- \_\_\_ v. Save the workspace (**Ctrl+S**).
- \_\_\_ w. Close the mapping editor.

### **Part 3: Wiring the data map into the AccountVerification process**

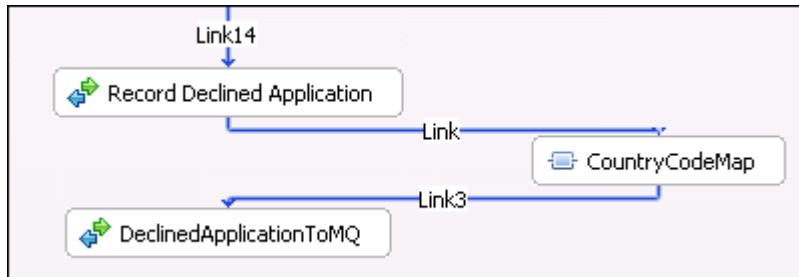
Now you wire the data map activity into the process just before the WebSphere MQ invoke. This action translates the country name to the numeric country code before it sends the data to the WebSphere MQ queue.

- \_\_\_ 1. Switch to the **AccountVerification** editor tab.

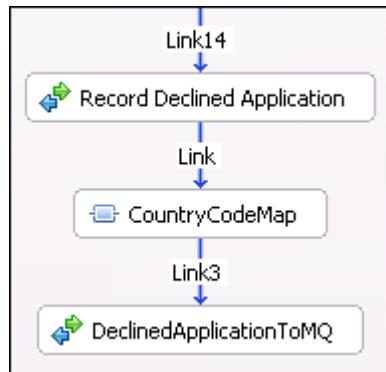
- \_\_ 2. In the **AccountVerification** process, locate the newly created **DataMap**.



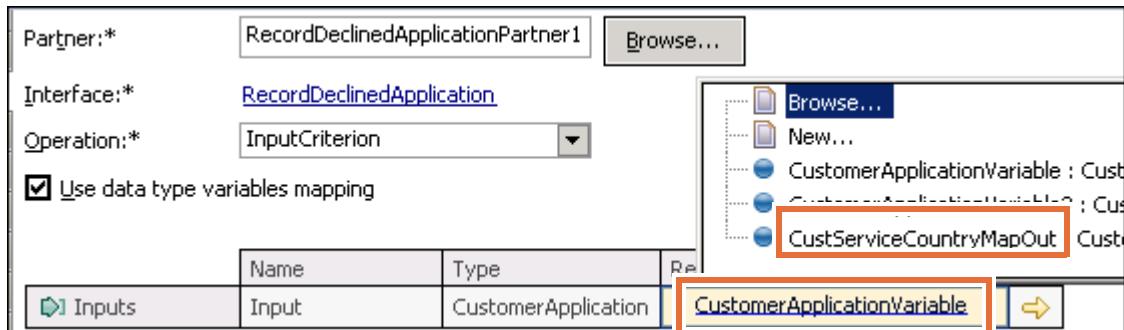
- \_\_ 3. Delete the link (link15) between **Record Declined Application** and **DeclinedApplicationToMQ** by right-clicking the link and selecting **Delete** from the menu.
- \_\_ 4. Rename the **DataMap** to **CountryCodeMap**.
- \_\_ a. Right-click **DataMap** and select the **Show In > Properties View** from the menu.
  - \_\_ b. Switch to the **Description** tab.
  - \_\_ c. Change the **Display Name** field to: **CountryCodeMap**
  - \_\_ d. Save the workspace.
- \_\_ 5. Link the **Record Declined Application** invoke activity to **CountryCodeMap**.
- \_\_ a. Right-click the **Record Declined Application** activity and choose **Add a link** from the menu.
  - \_\_ b. Click the **CountryCodeMap** activity to add the link.
  - \_\_ c. Accept the default link name.
- \_\_ 6. Similarly, link the **CountryCodeMap** to the invoke activity **DeclinedApplicationToMQ**.
- \_\_ a. Right-click the **CountryCodeMap** activity and choose **Add a link** from the menu.
  - \_\_ b. Click the **DeclinedApplicationToMQ** activity to add the link.
  - \_\_ c. Accept the default link name.



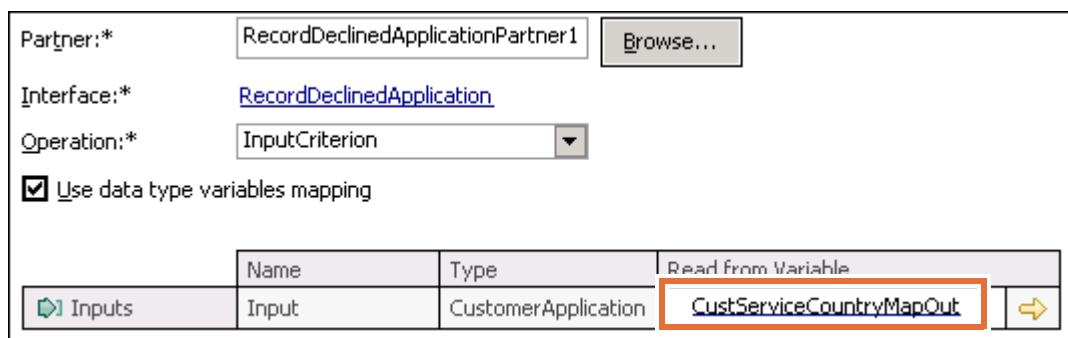
- 7. For readability, right-click the diagram, and choose **Arrange Parallel Activities Contents** from the menu.



- 8. Save the workspace.
- 9. Change the input variable that is used in the **DeclinedApplicationToMQ** invoke activity. Since the data mapping operation was inserted before the MQ invoke, you must change the input data that the MQ invoke uses. Change the input variable from **CustomerApplicationVariable** to **CustServiceCountryMapOut** (the new output variable you defined when you created the data map).
- Right-click **DeclinedApplicationToMQ** and choose **Show In > Properties View** from the menu.
  - In the **Properties** view, click the **Details** tab.
  - In the **Read From Variable** field, click **CustomerApplicationVariable**. When the list of variables is opened, click **CustServiceCountryMapOut**.



The updated properties show the new variable as the **Read from Variable** value.



- 10. Save the workspace (**Ctrl+S**).

- \_\_\_ 11. Close the BPEL editor tab.

The module is ready for testing.

## **Part 4: Testing the application**

You test the module by using a customer that has a high credit risk. Because of this risk, the application is declined. The process flow takes the “Application is declined” path, which invokes the Generate Decline and Record Declined Application activities. The CountryCodeMap activity is then invoked to translate the country code that the DeclinedApplicationToMQ activity follows. The DeclinedApplicationToMQ activity sends the message to a WebSphere MQ queue.

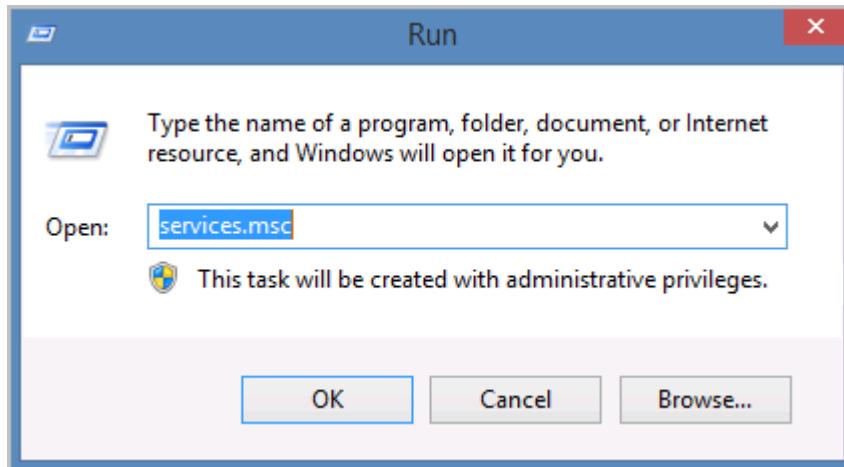
- \_\_\_ 1. Start the server (If it is not already running).
  - \_\_\_ a. In the **Servers** view, right-click **IBM Process Server v8.5.7 at localhost**.
  - \_\_\_ b. From the menu, choose **Start**.
  - \_\_\_ c. Wait for the server to start before proceeding. The **State** column shows **Started** and the Server Logs view shows the message `Server server1 open for e-business`.
- \_\_\_ 2. Deploy the modules to the server.
  - \_\_\_ a. In the **Servers** view, right-click **IBM Process Server v8.5.7 at localhost**.
  - \_\_\_ b. Select **Add and Remove** from the menu.
  - \_\_\_ c. Click **Add All** to add the modules to the **Configured** list.

- \_\_\_ d. Click **Finish**. The projects are deployed to the server. Wait for the deployment process to complete. This process might take several minutes. When complete, the state of the applications changes to **Started** in the Servers view. Additionally, in the **Server Logs** view, you see several messages that indicate the deployed applications started successfully, such as: Application started: FoundationModuleApp.

| Contents                                                       |
|----------------------------------------------------------------|
| WSVR0200I: Starting application: AccountOpeningUIEAR           |
| Development State Manager Decorator loaded.                    |
| WSVR0221I: Application started: AccountOpeningUIEAR            |
| WSVR0200I: Starting application: FoundationModuleApp           |
| WSVR0221I Application started: FoundationModuleApp             |
| ***** Start Display Current Environment *****                  |
| WSVR0200I: Starting application: IneligibleMediationServiceApp |
| WSVR0221I: Application started: IneligibleMediationServiceApp  |
| WSVR0200I: Starting application: CreditScoreServiceApp         |
| WSVR0221I: Application started: CreditScoreServiceApp          |
| WSVR0200I: Starting application: RouterMediationServiceApp     |
| WSVR0221I: Application started: RouterMediationServiceApp      |
| WSVR0200I: Starting application: AccountProcessingUIEAR        |
| WSVR0221I: Application started: AccountProcessingUIEAR         |
| WSVR0200I: Starting application: HumanTaskServicesApp          |
| WSVR0221I: Application started: HumanTaskServicesApp           |
| WSVR0200I: Starting application: FoundationServicesApp         |
| WSVR0221I: Application started: FoundationServicesApp          |

- \_\_\_ 3. Start the WebSphere MQ service.
- \_\_\_ a. On the Windows desktop, right-click the Windows logo icon and click **Run** from the menu.

- \_\_\_ b. Enter services.msc in the Run window and click OK. The **Services** window opens.



- \_\_\_ c. Locate the **IBM WebSphere MQ** service.

| Name                                    | Description             | Status  |
|-----------------------------------------|-------------------------|---------|
| Hyper-V Volume Shadow Copy Re...        | Coordinates the co...   |         |
| IBM Secure Shell Server for Windows     | IBM Secure Shell Se...  | Running |
| <b>IBM WebSphere MQ (Installation1)</b> | Provides startup an...  |         |
| IKE and AuthIP IPsec Keying Modu...     | The IKEEXT service ...  | Running |
| Interactive Services Detection          | Enables user notific... |         |
| Internet Connection Sharing (ICS)       | Provides network a...   |         |

- \_\_\_ d. If the service is not started already, right-click **IBM WebSphere MQ** and choose **Start** from the menu. The service is running when the **Status** column shows **Running**.

| Name                                    | Description             | Status         |
|-----------------------------------------|-------------------------|----------------|
| Hyper-V Volume Shadow Copy Re...        | Coordinates the co...   |                |
| IBM Secure Shell Server for Windows     | IBM Secure Shell Se...  | Running        |
| <b>IBM WebSphere MQ (Installation1)</b> | Provides startup an...  | <b>Running</b> |
| IKE and AuthIP IPsec Keying Modu...     | The IKEEXT service ...  | Running        |
| Interactive Services Detection          | Enables user notific... |                |

- \_\_\_ e. Minimize (but do not close) the Services window. You can use it later to stop the WebSphere MQ service.
- \_\_\_ f. Locate the **IBM WebSphere MQ** service.

| Name                                    | Description |
|-----------------------------------------|-------------|
| IBM Secure Shell Server for Windows     |             |
| <b>IBM WebSphere MQ (Installation1)</b> |             |
| IKE and AuthIP IPsec Keying Modules     |             |
| Interactive Services Detection          |             |

- \_\_\_ g. The service is running when the **Status** column shows **Running**. If the status is not **Running**, then right-click the service and select **Start** from the menu.

| Name                                    | Description             | Status         |
|-----------------------------------------|-------------------------|----------------|
| IBM Secure Shell Server for Windows     | IBM Secure Shell Se...  | Running        |
| <b>IBM WebSphere MQ (Installation1)</b> | Provides startup an...  | <b>Running</b> |
| IKE and AuthIP IPsec Keying Modules     | The IKEEXT service ...  | Running        |
| Interactive Services Detection          | Enables user notific... |                |

- \_\_\_ h. Minimize (but do not close) the Services control panel. You use it later to stop the WebSphere MQ service.
- \_\_\_ 4. Enter the initial test data through the Account Opening user interface by using **companyName** TestCo.

TestCo represents a company that has a high credit risk. As a result, the business process declines the application.

- \_\_\_ a. Start a Firefox browser session.
- \_\_\_ b. Enter the address: <https://localhost:9443/AccountOpeningUI/Login.jsp>  
The Login to Business User Client web page opens.
- \_\_\_ c. Use **Name** admin and **Password** web1sphere and then click **Login**.

**Login to Business User Client**

Enter user name and password, then click Login.

Name:

Password:

**Login**

The New Business Case page opens.

- \_\_ d. Click **New**.

The screenshot shows the 'Business User Client' interface. In the 'Business Case' section, there is a button labeled '→ New' which is highlighted with a red border. Below the button, a tooltip-like message reads: 'Select this to view a list of all the tasks that you can use to create a business case. criteria, you can provide additional information.'

The New Business Cases page opens.

- \_\_ e. Click **CreateApplication**.

The screenshot shows the 'Business Cases > New' page. Under the 'Task' section, there is a button labeled '→ CreateApplication' which is highlighted with a red border.

The CreateApplication Input Data page opens.

- \_\_ f. Enter **TestCo** in the **companyName** field. Leave the other fields blank.

The screenshot shows the 'Business Cases > New > CreateApplication' input data page. The 'Input Data' section contains several fields: accountNumber, applicationDate, applicationDecision (checkbox), comments, companyName, contactFirstName, and contactLastName. The 'companyName' field contains the value 'TestCo', which is highlighted with a red border.

- \_\_\_ g. Click **Create** at the bottom of the page.

The New Business Cases page is opened again. The application processes the input data up to the Request More Documentation human task.

- \_\_\_ h. Minimize the browser window.
- \_\_\_ 5. Using the Business Process Choreographer Explorer client, claim the Request More Documentation task.
- \_\_\_ a. Switch to the **Servers** view in IBM Integration Designer.
  - \_\_\_ b. Right-click **IBM Process Server v8.5.7 at localhost** and choose **Launch > Business Process Choreographer Explorer** from the menu.
  - \_\_\_ c. If the security alert window prompts you, click **Yes** to proceed.
  - \_\_\_ d. In the Business Process Choreographer Explorer login page, use **User Name** `admin` and **Password** `web1sphere` and click **Login**. The My To-dos page is opened.
  - \_\_\_ e. A task is shown in the task list for you to claim. If the task does not show, click **Refresh**. Click the **Request More Documentation** link.

| <input type="checkbox"/> Priority | Task Name                  | State | Kind       | Owner |
|-----------------------------------|----------------------------|-------|------------|-------|
| <input type="checkbox"/> 5        | Request More Documentation | Ready | To-do Task |       |

The Task Instance page opens.

- \_\_\_ f. Click **Work on** to claim the task.

The screenshot shows the 'Task Instance' page. At the top, there is a header with the title 'Task Instance'. Below the header, a message says 'Use this page to display information about the task and, option...'. There is a row of buttons: 'Work on' (highlighted with a red box), 'Terminate', 'Suspend', 'Restart', and 'Work Items'. Underneath these buttons is a section titled 'Task Description'. Below that, there is a table with three rows:

|             |                                   |
|-------------|-----------------------------------|
| Task Name   | Request More Documentation        |
| Description | Request More Documentation sta... |
| Reason      | Administrator, Originator         |

The Task Message page opens.

- \_\_\_ g. Scroll to the **Task Output Message** section, and click **Edit Source** at the bottom of the section.



The **task input** and **task output** message fields are shown.

- \_\_\_ h. Scroll to the bottom of the page, to the **Task Output Message** section.  
 \_\_\_ i. In Windows Explorer, go to `C:\labfiles\Support Files\EX9`.  
 \_\_\_ j. Open `EX9_Test_Data1A.txt` in a text editor.  
 \_\_\_ k. Copy the text from the file (use **Ctrl+A** to select all of the lines, and **Ctrl+C** to copy). Then, paste it (use **Ctrl+V**) over the existing text in the **Task Output Message Source View** window.

Task Output Message

**Source View**

```
<?xml version="1.0" encoding="UTF-8"?>
<p:InputCriterionResponse xsi:type="p:InputCriterionResponse_.1"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:p="http://FoundationLibrary/RequestMoreDocumentation">
 <Output>
 <accountNumber>TEST001</accountNumber>
 <applicationDate>Jul 31, 2014</applicationDate>
 <applicationDecision>false</applicationDecision>
 <comments>Complete</comments>
 <companyName>TestCo</companyName>
 <contactFirstName>Jane</contactFirstName>
 <contactLastName>Doe</contactLastName>
 <contactPhoneNumber>872-555-9915</contactPhoneNumber>
 <creditRating>C</creditRating>
 <creditReportNeeded>true</creditReportNeeded>
 <creditRisk>HIGH</creditRisk>
 <creditScore>1</creditScore>
 <customerCity>Helsinki</customerCity>
 <customerCountry>FINLAND</customerCountry>
```

- \_\_\_ l. Click **Confirm**. You return to the top of the Task Message page.

- \_\_\_ m. Scroll to **Task Output Message** again, and verify that the **Comments** field contains the value **Complete**.

**Task Output Message**

| <b>Form View</b>    |                                 |
|---------------------|---------------------------------|
| Output              |                                 |
| accountNumber       | TEST001                         |
| applicationDate     | Jul 31, 2014                    |
| applicationDecision | <input type="checkbox"/> Remove |
| comments            | Complete                        |

- \_\_\_ n. Scroll to the top of the page again, and click **Complete** to finish the task.
- \_\_\_ o. Leave the Windows Explorer session open, but close `EX9_Test_Data1A.txt`.
- \_\_\_ 6. Set the value of the `applicationDecision` field to `false` to decline the application.
- \_\_\_ a. On the left side of the **My To-dos** page, scroll to the **Task Instances** section and click **My To-dos**. The list of tasks is updated. A **Final Application Review** task is present.
- \_\_\_ b. Click the check box to the left of the **Final Application Review** task, and click **Work On**.

**My To-dos**

Use this page to work on tasks that are assigned to you. [i](#)

|                                     |            |                          |         |                          |
|-------------------------------------|------------|--------------------------|---------|--------------------------|
| <b>Work on</b>                      | Release    | Transfer                 | Start   | Change Business Category |
| <input type="checkbox"/>            | Priority ◊ | Task Name ◊              | State ◊ | Kind ◊                   |
| <input checked="" type="checkbox"/> | 5          | Final Application Review | Ready   | To-do Task               |

- \_\_\_ c. Scroll to the **Task Output Message** section, and click **Edit Source** at the bottom of the section.
- \_\_\_ d. Switch to the Windows Explorer session.
- \_\_\_ e. Open the file `EX9_Test_Data1B.txt` in a text editor.
- \_\_\_ f. Copy the text from the file (use **Ctrl+A** to select all of the lines, and **Ctrl+C** to copy). Then, paste it (use **Ctrl+V**) over the existing text in the **Task Output Message Source View** window.
- \_\_\_ g. Click **Confirm**. You are returned to the form view.
- \_\_\_ h. Close `EX9_Test_Data1B.txt`.

- \_\_\_ i. Verify that the value of **applicationDecision** in **Task Output Message** is set to `false` (the check box is not checked).

Task Output Message

| Form View           |                                 |
|---------------------|---------------------------------|
| Output              |                                 |
| accountNumber       | TEST001                         |
| applicationDate     | Jul 31, 2014                    |
| applicationDecision | <input type="checkbox"/> Remove |
| comments            | Complete                        |
| companyName         | TestCo                          |

- \_\_\_ j. Notice that the value of **customerCountry** is `FINLAND`.

|                 |         |
|-----------------|---------|
| customerCountry | FINLAND |
|-----------------|---------|

- \_\_\_ k. At the top of the page, click **Complete**. The remainder of the AccountVerification flow runs, and the application is denied. The data map is then invoked, which in turn invokes the relationship that the Customer Service system uses to convert the country name `FINLAND` to country code `246`. You verify that in a subsequent step.

- \_\_\_ 7. Log out of Business Process Choreographer Explorer and close the tab.

## Part 5: Reviewing the test results

You now review the results of the test to verify that the correctly mapped country code is included in the message sent to WebSphere MQ for processing by the Customer Service system.

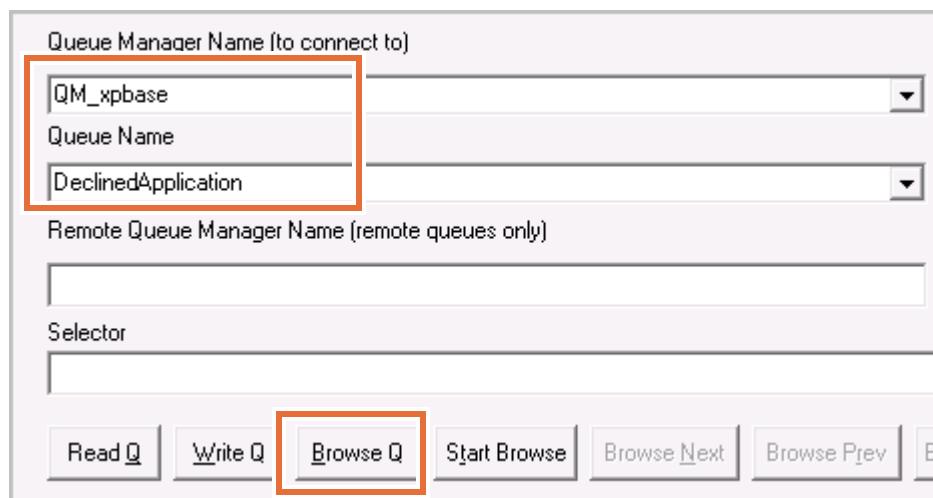
- \_\_\_ 1. Check the execution flow in the server logs.
- \_\_\_ a. Switch to the **Server Logs** view.
- \_\_\_ b. Review the log messages. You see that the Record Declined Application service invocation indicates that the application was declined.

|                                                                                               |
|-----------------------------------------------------------------------------------------------|
| [Java] Determine Applicant Eligibility - begins                                               |
| [Java] Determine Applicant Eligibility - ends                                                 |
| [Java] Generate Decline - begins                                                              |
| <b>[Java] Generate Decline - Account for customer TestCo was declined and the credit r...</b> |
| [Java] Generate Decline - ends                                                                |
| [Java] Record Declined Application - begins                                                   |
| [Java] Record Declined Application - ends                                                     |

- \_\_\_ 2. Use the **RFHUtil** program to browse the message.

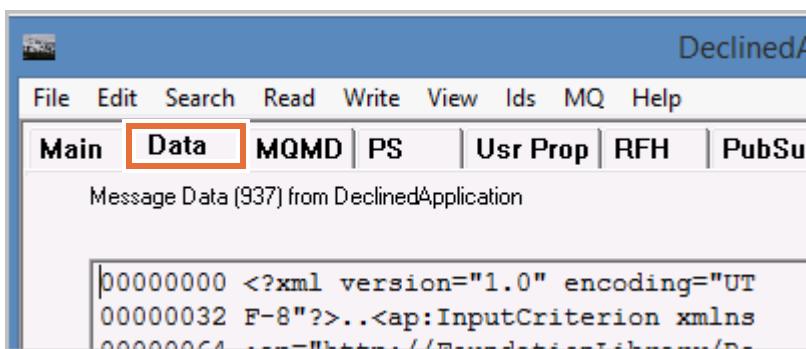
RFHUtil is a utility that can be used to browse WebSphere MQ queues and also the JMS queues. It is available for download as IBM MQ SupportPac IH03.

- \_\_ a. On the Windows desktop, double-click the RFHUtil shortcut. You can also browse to C:\Tools\RFHUtil and double-click **RFHUTIL.EXE**.
- \_\_ b. The main panel of RFHUtil opens. Click the **Main** tab unless it is already selected.
- \_\_ c. For the **Queue Manager Name** field, choose **QM\_xpbase** from the list.
- \_\_ d. For the **Queue Name** field, choose **DeclinedApplication** from the list.
- \_\_ e. Click **Browse Q**.

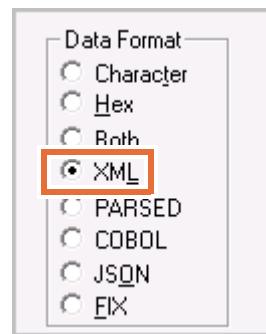


The message is retrieved from the **DeclinedApplication** queue.

- \_\_ f. Click the **Data** tab at the top of the pane to view the message.



- \_\_ g. To change the data into a more meaningful format, select the **XML** option under the **Data Format** section on the right side of the pane.



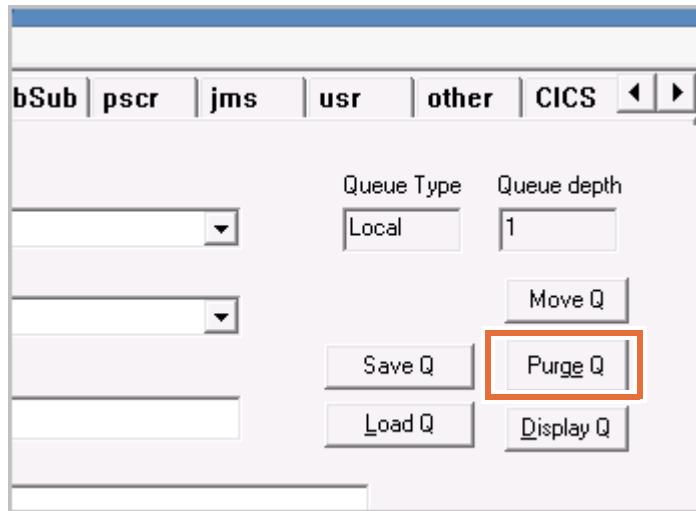
- \_\_ h. Notice the value of **customerCountry** in the message. It is converted from the value FINLAND to the value 246 by the static relationship.

```
<?xml version="1.0" encoding="UTF-8"?>
<ap:InputCriterion xmlns:ap="http://FoundationLibrary/RecordDecl">
<Input>
 <accountNumber>TEST001</accountNumber>
 <applicationDate>Jul 31, 2014</applicationDate>
 <applicationDecision>false</applicationDecision>
 <comments>Complete</comments>
 <companyName>TestCo</companyName>
 <contactFirstName>Jane</contactFirstName>
 <contactLastName>Doe</contactLastName>
 <contactPhoneNumber>872-555-9915</contactPhoneNumber>
 <creditRating>C</creditRating>
 <creditReportNeeded>true</creditReportNeeded>
 <creditRisk>HIGH</creditRisk>
 <creditScore>1</creditScore>
 <customerCity>Helsinki</customerCity>
 <customerCountry>246</customerCountry>
 <eligibleApplication>true</eligibleApplication>
 <ineligibleReason>Uncertain collateral</ineligibleReason>
 <pricingCode>57</pricingCode>
 <pricingScore>11</pricingScore>
 <productName>Chairs</productName>
 <requestAccountAmount>50000</requestAccountAmount>
</Input>
</ap:InputCriterion>
```

- \_\_ 3. When you are done browsing the queue, delete the message.

- \_\_ a. Click the **Main** tab.

- \_\_\_ b. Click **Purge Q**.



- \_\_\_ c. Click **Yes** on the purge confirmation window.  
\_\_\_ 4. Close RFHUtil.

## **Part 6: Reviewing the relationship data by using the Relationship Manager**

The roles and instance data for relationships are stored in tables in the IBM Process Server runtime environment. You can use a tool that is called the Relationship Manager to examine and modify relationship information.

- \_\_\_ 1. Start the Relationship Manager.
  - \_\_\_ a. Switch to the **Servers** view.
  - \_\_\_ b. Right-click **IBM Process Server v8.5.7 at localhost** and click **Launch > Relationship Manager** from the menu. The console login pane opens.
  - \_\_\_ c. If a security alert window prompts you, click **Yes** to proceed.
  - \_\_\_ d. Use **User ID** `admin` and **Password** `web1sphere` and click **Log in**. The Integrated Solutions Console menu is opened.

- \_\_ e. In the **Task** panel at the left, expand **Integration Applications** and then click **Relationship Manager**.



The relationship manager opens.

- \_\_ 2. Show the characteristics of the **CountryCode** relationship.  
\_\_ a. Under the **Relationships** heading, click the **Relationships** link.

The screenshot shows the 'Relationship Manager' page. At the top, it says 'Cell=PSCell1, Profile=qbpmaps'. Below that is a 'Relationship Manager' header. The main content area has a heading 'Relationship Manager' and a sub-instruction 'The relationship manager is used to query and manage'. Below this is a 'Relationships' section with three circular icons. At the bottom, it lists 'Following are the Relationship services MBeans on your System' and shows 'PSCell1:Node1:server' followed by a 'Relationships' link, which is highlighted with a red box.

The list of relationships that are defined to the server is shown. Notice the actions that can be done, including importing new instance data, creating a relationship, querying on an existing relationship, and other actions.

- \_\_ b. Click the radio button in the **Select** column and then click **Details**.

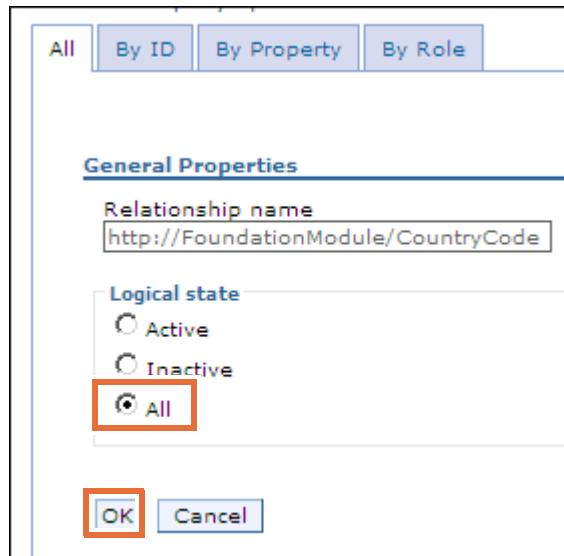


The properties of the relationship are shown, including the participating roles.

| General Properties                                 |                                                                  |                     |
|----------------------------------------------------|------------------------------------------------------------------|---------------------|
| <b>Relationship name</b>                           | <input type="text" value="http://FoundationModule/CountryCode"/> |                     |
| <b>Display Name</b>                                | <input type="text" value="CountryCode"/>                         |                     |
| <b>Role schema information</b>                     |                                                                  |                     |
| Name                                               | Display name                                                     | Object Name         |
| <a href="#">CountryCode_CustomerApplication</a>    | CountryCode_CustomerApplication                                  | CustomerApplication |
| <a href="#">CountryCode_CustomerApplication0</a>   | CountryCode_CustomerApplication0                                 | CustomerApplication |
| <b>Property values</b>                             |                                                                  |                     |
| <input type="text" value="No properties defined"/> |                                                                  |                     |

- \_\_ c. At the bottom of the panel, click **Back** to return to the previous page.
- \_\_ 3. Open the instance data of the **CountryCode** relationship.
- \_\_ a. Click the radio button in the **Select** column, and then click **Query**. The query relationship panel opens.

- \_\_ b. On the **All** tab, accept the default **All** for **Logical state** and click **OK**.

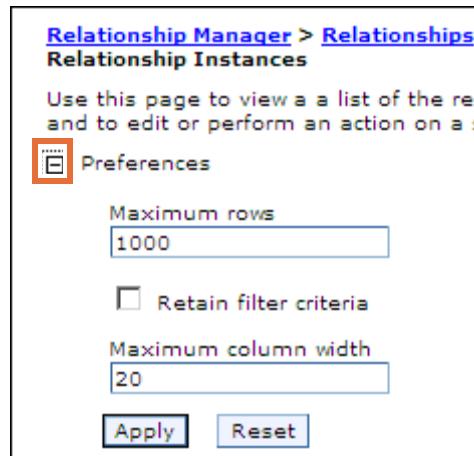


The instance data is listed in order by instance ID.



### Note

The **Preferences** menu controls the number of rows that are opened. Expand it to review or change the current settings.



- \_\_\_ c. Click the radio button in the **Select** column to the left of **Relationship instance ID 1** and then click **Details**. The role instance data is shown. Notice the values in the **Key Attribute** column.

**General Properties**

Relationship name  
http://FoundationModule/CountryCode

Relationship instance ID  
1

**Property values**

No properties defined

**Role Instances**

| CountryCode_CustomerApplication0 |                   |               |                   |                |
|----------------------------------|-------------------|---------------|-------------------|----------------|
| Select                           | ID                | Logical State | Key Attributes    | Property Value |
| <input checked="" type="radio"/> | <a href="#">1</a> | Active        | customerCountry=4 |                |

[Create](#) [Delete](#)

| CountryCode_CustomerApplication  |                   |               |                             |              |
|----------------------------------|-------------------|---------------|-----------------------------|--------------|
| Select                           | ID                | Logical State | Key Attributes              | Property Val |
| <input checked="" type="radio"/> | <a href="#">1</a> | Active        | customerCountry=AFGHANISTAN |              |

- \_\_\_ d. Click the **Query Relationship: CountryCode** link to return to the query relationship panel.

**Relationship Manager**

Relationship Manager > Relationships > **Query Relationship: CountryCode** > Relationship Instances > Relationship

Use this page to view detailed information for the selected relationship instance and to edit or perform an action on the relationship instance.

- \_\_\_ e. Select the **By Role** tab. From here, you can query by instance data values.
- \_\_\_ f. For **Role Name**, select [http://FoundationModule/CountryCode\\_CustomerApplication](#) from the list.

- \_\_ g. Enter SPAIN in the **Value** field of the **Key Attributes** section, and then click **OK**.

| Key attributes  |       |
|-----------------|-------|
| Name            | Value |
| customerCountry | SPAIN |

Instance ID **203** is returned.

| Relationship instance ID |                     |
|--------------------------|---------------------|
| <input type="radio"/>    | <a href="#">203</a> |
| Total 1                  |                     |

- \_\_ h. To view the ISO numeric value that corresponds with the ISO alphanumeric value **SPAIN**, click the **Query Relationship: CountryCode** link to return to the query relationship panel, and select the **By ID** tab.
- \_\_ i. Enter 203 in both **Starting ID** and **Ending ID** fields. Click **OK**.
- \_\_ j. Click the radio button in the **Select** column and then click **Details**. The role instance data is shown. Notice that the ISO numeric value that corresponds with **SPAIN** is 724.

#### Role Instances

##### CountryCode\_CustomerApplication0

| Select                           | ID                | Logical State | Key Attributes      | Property Values |
|----------------------------------|-------------------|---------------|---------------------|-----------------|
| <input checked="" type="radio"/> | <a href="#">1</a> | Active        | customerCountry=724 |                 |

[Create](#) [Delete](#)

##### CountryCode\_CustomerApplication

| Select                           | ID                | Logical State | Key Attributes        | Property Values |
|----------------------------------|-------------------|---------------|-----------------------|-----------------|
| <input checked="" type="radio"/> | <a href="#">1</a> | Active        | customerCountry=SPAIN |                 |

- \_\_ 4. Click **Logout** and close the Relationship Manager tab.

- \_\_\_ 5. If you intend to proceed to the next optional section, leave the applications deployed and the server still running. Otherwise, remove the applications.
  - \_\_\_ a. In the **Servers** view, right-click **IBM Process Server v8.5.7 at localhost** and choose **Add and Remove** from the menu.
  - \_\_\_ b. Click **Remove All** and click **Finish**.
- \_\_\_ 6. If you are not proceeding to the optional section, close IBM Integration Designer.

## **Part 7: (Optional) Testing with an invalid country code**

If you want, you can test the module by using the same customer, but with an alphanumeric country code that does not exist in the instance data for the lookup relationship. Again, the application is declined, and the process flow takes the “Application is declined” path. This change invokes the Generate Decline and Record Declined Application services. The CountryCodeMap activity is then invoked to translate the country code. Then, the DeclinedApplicationToMQ activity, which sends the message to a WebSphere MQ queue, follows the DeclinedApplicationToMQ activity.

- \_\_\_ 1. Enter the initial test data through the Account Opening user interface.
  - \_\_\_ a. Maximize the Firefox browser session that you used previously, or open a new browser session if you closed it.
  - \_\_\_ b. Since your session was timed out, enter the address:  
<https://localhost:9443/AccountOpeningUI/Login.jsp>  
 The Login to Business User Client web page opens.
  - \_\_\_ c. Use **Name** admin and **Password** web1sphere and click **Login**.



The New Business Case page opens.

- \_\_ d. Click **New**.

The screenshot shows the 'Business User Client' interface. In the 'Business Case' section, there is a button labeled '+ New' which is highlighted with a red border. A tooltip below it says: 'Select this to view a list of all the tasks that you can use to create a business case. You can provide additional information.'

The New Business Cases page opens.

- \_\_ e. Click **CreateApplication**.

The screenshot shows the 'Business Cases > New' page. Under the 'Task' section, there is a button labeled '+ CreateApplication' which is highlighted with a red border.

The CreateApplication Input Data page opens.

- \_\_ f. Enter **TestCo** in the **companyName** field.

The screenshot shows the 'Business Cases > New > CreateApplication' input data page. The 'Input Data' section contains several fields: accountNumber, applicationDate, applicationDecision (checkbox), comments, companyName (text input field containing 'TestCo'), contactFirstName, and contactLastName. The 'companyName' field is highlighted with a red border.

- \_\_\_ g. Leave the other fields blank, and click **Create** at the bottom of the page. The **New Business Cases** page is opened again. The application processes the input data up to the Request More Documentation human task.
  - \_\_\_ h. Minimize the browser window.
- \_\_\_ 2. Using the Business Process Choreographer Explorer client, claim the Request More Documentation task.
- \_\_\_ a. Switch to the **Servers** view in IBM Integration Designer.
  - \_\_\_ b. Right-click **IBM Process Server v8.5.7 at localhost** and select **Launch > Business Process Choreographer Explorer** from the menu.
  - \_\_\_ c. If a security alert window prompts you, click **Yes** to proceed.
  - \_\_\_ d. In the Business Process Choreographer Explorer login page, use **User Name** `admin` and **Password** `web1sphere` and click **Login**. The My To-dos page is opened.
  - \_\_\_ e. A task is shown in the task list for you to claim. Click the **Request More Documentation** link.

| <input type="checkbox"/> Priority ◁ | Task Name ◁                | State ◁ | Kind ◁     | Owner ◁ |
|-------------------------------------|----------------------------|---------|------------|---------|
| <input type="checkbox"/> 5          | Request More Documentation | Ready   | To-do Task |         |

The Task Instance page opens.

- \_\_\_ f. Click **Work On** to claim the task.

**Task Instance**

Use this page to display information about the task and, option

**Work on** **Terminate** **Suspend** **Restart** **Work Items**

**Task Description**

|             |                                |
|-------------|--------------------------------|
| Task Name   | Request More Documentation     |
| Description | Request More Documentation sta |
| Reason      | Administrator, Originator      |

The Task Message page opens.

- \_\_\_ g. Scroll to the **Task Output Message** section, and click **Edit Source** at the bottom of the section.

|                      |
|----------------------|
| pricingCode          |
| pricingScore         |
| productName          |
| requestAccountAmount |

**Edit Source**

The Task Input Message and Task Output Message sections are opened.

- \_\_ h. Scroll to the bottom of the page to the **Task Output Message** section.
- \_\_ i. In Windows Explorer, go to C:\labfiles\Support Files\EX9.
- \_\_ j. Open EX9\_Test\_Data2A.txt in a text editor.
- \_\_ k. Copy the text from the file (use **Ctrl+A** to select all of the lines, and **Ctrl+C** to copy). Then, paste it (use **Ctrl+V**) over the existing text in the **Task Output Message Source View** window.

Task Output Message

**Source View\***

```
<?xml version="1.0" encoding="UTF-8"?>
<p:InputCriterionResponse xsi:type="p:InputCriterionResponse_
_
xm...></p:InputCriterionResponse>
<Output>
<accountNumber>TEST001</accountNumber>
<applicationDate>Jul 31, 2014</applicationDate>
<applicationDecision>false</applicationDecision>
<comments>Complete</comments>
<companyName>TestCo</companyName>
<contactFirstName>Jane</contactFirstName>
<contactLastName>Doe</contactLastName>
<contactPhoneNumber>872-555-9915</contactPhoneNumber>
<creditRating>C</creditRating>
<creditReportNeeded>true</creditReportNeeded>
<creditRisk>HIGH</creditRisk>
<creditScore>1</creditScore>
<customerCity>Belgrade</customerCity>
<customerCountry>YUGOSLAVIA</customerCountry>
```

- \_\_ l. Click **Confirm**. You are returned to the top of the Task Message page.
- \_\_ m. Scroll to **Task Output Message** again, and verify that the **comments** field contains the value **Complete**.

Task Output Message

**Form View**

| Output | accountNumber       | TEST001                         |
|--------|---------------------|---------------------------------|
|        | applicationDate     | Jul 31, 2014                    |
|        | applicationDecision | <input type="checkbox"/> Remove |
|        | comments            | Complete                        |

- \_\_ n. Scroll to the top of the page again, and click **Complete** to finish the task.
- \_\_ o. Leave the Windows Explorer session open, but close EX9\_Test\_Data2A.txt.

- \_\_\_ 3. Set the value of the `applicationDecision` field to `false` to decline the application.
  - \_\_\_ a. On the left side of the **My To-dos** page, scroll to the **Task Instances** section and click **My To-dos**. The list of tasks is updated. A **Final Application Review** task is present.
  - \_\_\_ b. Click the check box to the left of the **Final Application Review** task, and click **Work On**.

**My To-dos**

Use this page to work on tasks that are assigned to you. [i]

[Work on] Release Transfer Start Change Business Category

[ ] Priority ◊ Task Name ◊ State ◊ Kind ◊

[ ] 5 Final Application Review Ready To-do Task

- \_\_\_ c. Scroll to the **Task Output Message** section, and click **Edit Source** at the bottom of the section.
- \_\_\_ d. Switch to Windows Explorer.
- \_\_\_ e. Open `EX9_Test_Data2B.txt` in a text editor.
- \_\_\_ f. Copy the text from the file (use **Ctrl+A** to select all of the lines, and **Ctrl+C** to copy). Then, paste it (use **Ctrl+V**) over the existing text in the **Task Output Message Source View** window.
- \_\_\_ g. Click **Confirm**. You are returned to the form view.
- \_\_\_ h. Close `EX9_Test_Data2B.txt`.
- \_\_\_ i. Verify that the value of `applicationDecision` is set to `false` (the check box is not checked).

**Task Output Message**

**Form View**

| Output              | accountNumber                   | TEST001 |
|---------------------|---------------------------------|---------|
| applicationDate     | Jul 31, 2014                    |         |
| applicationDecision | <input type="checkbox"/> Remove |         |
| comments            | Complete                        |         |
| companyName         | TestCo                          |         |

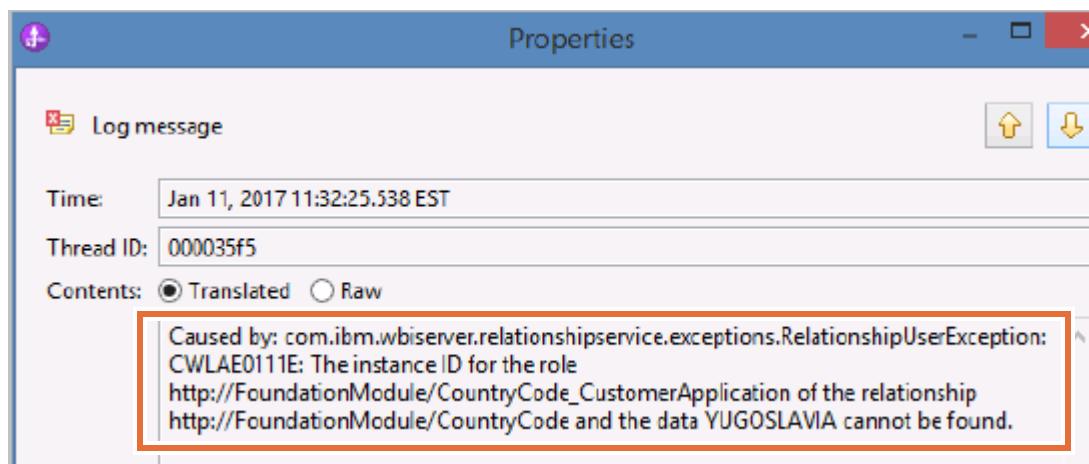
- \_\_\_ j. Notice that the value of **customerCountry** is **YUGOSLAVIA**.

|                 |            |
|-----------------|------------|
| customerCountry | YUGOSLAVIA |
|-----------------|------------|

- \_\_\_ k. At the top of the page, click **Complete**. The remainder of the AccountVerification flow runs, and the application is denied. This action causes the data map to be invoked. However, unlike the previous test case, the **customerCountry** value YUGOSLAVIA does not exist in the instance data table. This change causes a runtime exception to occur when the static lookup relationship (step 14 of the map) is invoked during the data map.

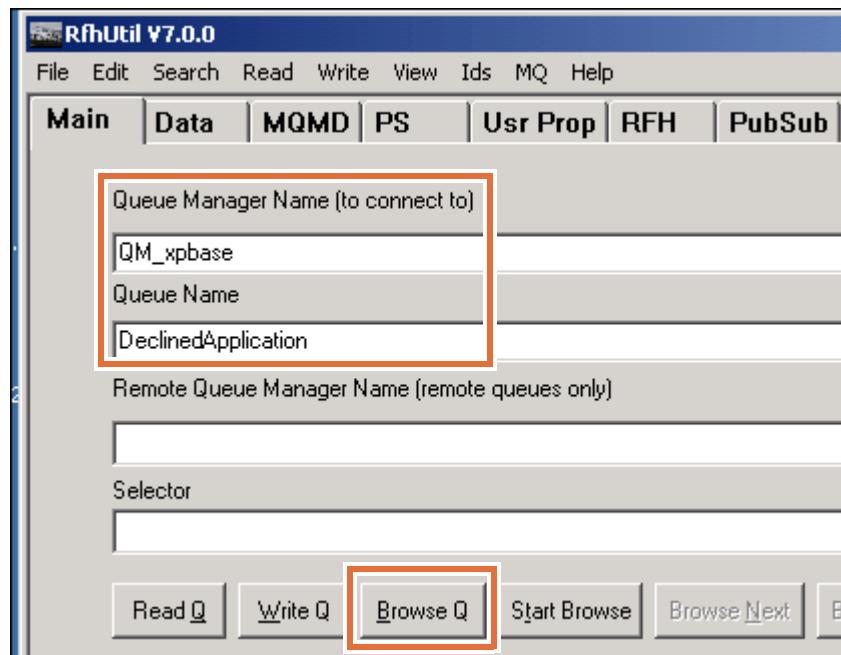
|          |                                                                                             |
|----------|---------------------------------------------------------------------------------------------|
| 0000008f | [Java] Generate Decline - begins                                                            |
| 0000008f | [Java] Generate Decline - Account for customer TestCo was declined and the credit risk w... |
| 0000008f | [Java] Generate Decline - ends                                                              |
| 0000008a | [Java] Record Declined Application - begins                                                 |
| 0000008a | [Java] Record Declined Application - ends                                                   |
| 0000008a | com.ibm.wbiserver.relationshipservice.impl.LookupAPIImpl.staticLookup(Object value, Rel...  |
| 0000008a | CustServiceCountryCode Transformation #14() CWLAS0015E: The Static Lookup transfor...       |

If you examine the detail of the message, the reason for the exception is stated in Server Logs. Double-click the Exception to see the detailed error message.

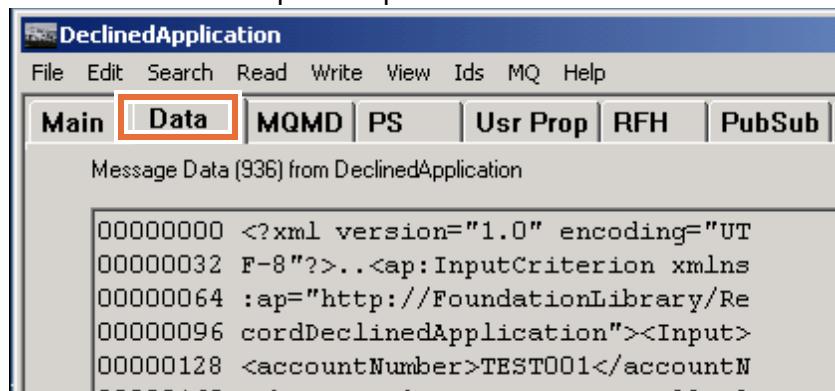


- \_\_\_ 4. Use the RFHUtil program to browse the message.
- \_\_\_ a. On the Windows desktop, double-click the RFHUtil shortcut. You can also browse to C:\Tools\RFHUtil and double-click RFHUTIL.EXE.
- \_\_\_ b. The main panel of RFHUtil opens. Click the **Main** tab unless it is already selected.
- \_\_\_ c. For the **Queue Manager Name** field, select QM\_xpbase from the selection list.
- \_\_\_ d. For the **Queue Name** field, select DeclinedApplication from the selection list.

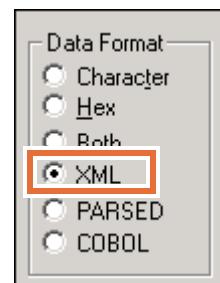
- \_\_ e. Click **Browse Q**.



- \_\_ f. Click the **Data** tab at the top of the pane.



- \_\_ g. To change the data into a more meaningful format, select the **XML** option under the **Data Format** section on the right side of the pane.



- \_\_\_ h. Notice that the body of the message is incomplete; the last element that is shown in the message is `customerCity`. The runtime exception that occurred when the relationship lookup failed caused the construction of the output message to stop, but the message is still passed to the **DeclinedApplicationToMQ** activity.

```
<?xml version="1.0" encoding="UTF-8"?>
<ap:InputCriterion xmlns:ap="http://FoundationLibrary/RecordDeclinedApplication">
 <Input>
 <accountNumber>TEST001</accountNumber>
 <applicationDate>Jul 31, 2014</applicationDate>
 <applicationDecision>false</applicationDecision>
 <comments>Complete</comments>
 <companyName>TestCo</companyName>
 <contactFirstName>Jane</contactFirstName>
 <contactLastName>Doe</contactLastName>
 <contactPhoneNumber>872-555-9915</contactPhoneNumber>
 <creditRating>C</creditRating>
 <creditReportNeeded>true</creditReportNeeded>
 <creditRisk>HIGH</creditRisk>
 <creditScore>1</creditScore>
 <customerCity>Belgrade</customerCity>
 </Input>
</ap:InputCriterion>
```

- \_\_\_ 5. When you are done browsing the queue, delete the message.
- \_\_\_ a. Click the **Main** tab.
  - \_\_\_ b. Click **Purge Q**.
  - \_\_\_ c. Click **Yes** on the purge confirmation window.
- \_\_\_ 6. Close the RFHUtil window.
- \_\_\_ 7. Click **Logout** and close the Business Process Choreographer Explorer tab.
- \_\_\_ 8. Remove the applications from the server and (optionally) stop the server.
- \_\_\_ a. In the **Servers** view, right-click **IBM Process Server v8.5.7 at localhost** and choose **Add and Remove** from the menu.
  - \_\_\_ b. Click **Remove All** and click **Finish**.
- \_\_\_ 9. Close the IBM Integration Designer. Click **File > Exit**, or click **X** in the upper-right corner.
- \_\_\_ 10. Stop the IBM MQ service.
- \_\_\_ a. Maximize the Services console.
  - \_\_\_ b. Locate the **IBM WebSphere MQ** service.
  - \_\_\_ c. Right-click the service and select **Stop** from the menu. The service is stopped when the **Status** column is empty.
  - \_\_\_ d. Close the Services control panel.

## End of exercise

# Exercise 10. Implementing a mediation flow

## Estimated time

01:00

## Overview

In this exercise, you implement a mediation flow with multiple mediation primitives.

## Objectives

After completing this exercise, you should be able to:

- Implement a service aggregation flow in a mediation module with multiple mediation primitives, including fan out, fan-in, service invoke, XSL transformation, message element setter, and custom mediation
- Test a mediation module in the IBM Integration Designer integrated test environment

## Introduction

Any service can replace any other service that does the same processing as the replacing service. Replacing a service should not require any changes to the existing caller. For example, assume that application A starts the service of application B. If application C is functionally equivalent to application B, then application A must be able to start application C instead of application B, without the need to change application A in any manner.

Recall that in the scenario that is used for the exercises in this course, a credit check is run for all eligible credit applications. In this exercise, you replace the existing credit check application with one that queries three different credit report services. The application then returns the lowest of the three obtained credit scores to the calling service. You implement the new credit score inquiry service as a mediation module. This module uses an aggregation flow, which starts the three credit report services, determines the lowest score, and then returns that score to the account verification module.

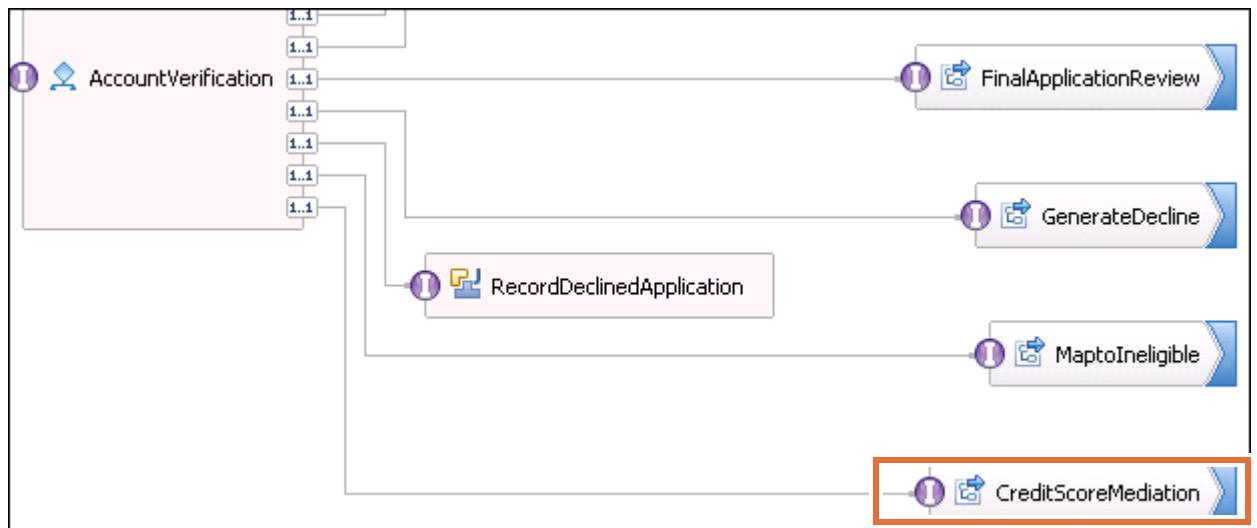
## Requirements

Completing the exercise requires a lab environment that includes the exercise support files, Mozilla Firefox, IBM Business Process Manager V8.5.7 Advanced, IBM DB2 V10.1, and the IBM Process Server V8.5.7 test environment.

## Part 1: Reviewing the workspace

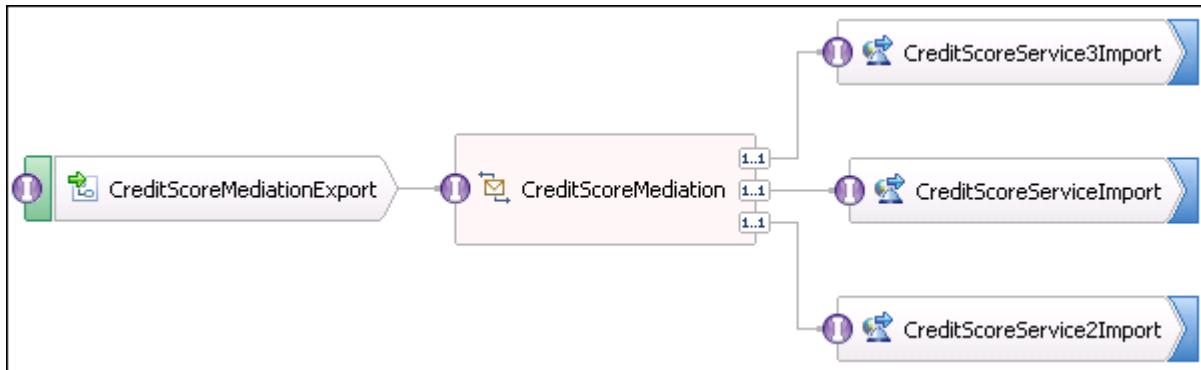
- 1. Open the Exercise 10 workspace.
  - a. On your desktop, open the folder that is labeled **Exercise Shortcuts**.
  - b. Double-click the shortcut that is labeled **Exercise 10**. IBM Integration Designer starts.
  - c. Wait for the workspace build to complete; it might take a few minutes.
  - d. Close the **Getting Started** tab.
- 2. Review the assembly diagram for `FoundationModule`.
  - a. In the **Projects** section of the **Business Integration** view, expand `FoundationModule`.
  - b. Double-click **Assembly Diagram**.

The assembly diagram opens in the editor pane.



The `CreditScoreMediation` import replaced the `CreditScoreServiceImport` from the scenario. This import is how the `AccountVerification` process starts the mediation module, which determines the lowest credit score for the customer application that is being processed.

- \_\_\_ 3. Review the assembly diagram for the `CreditScoreMediationService`.
  - \_\_\_ a. In the **Projects** section of the **Business Integration** view, expand `CreditScoreMediationService`, and double-click **Assembly Diagram**. The assembly diagram opens in the editor pane.



`CreditScoreMediation` accepts a company name and then starts the three credit score services. Each of those services returns a credit score for the company, and the `CreditScoreMediation` returns the lowest of those scores.

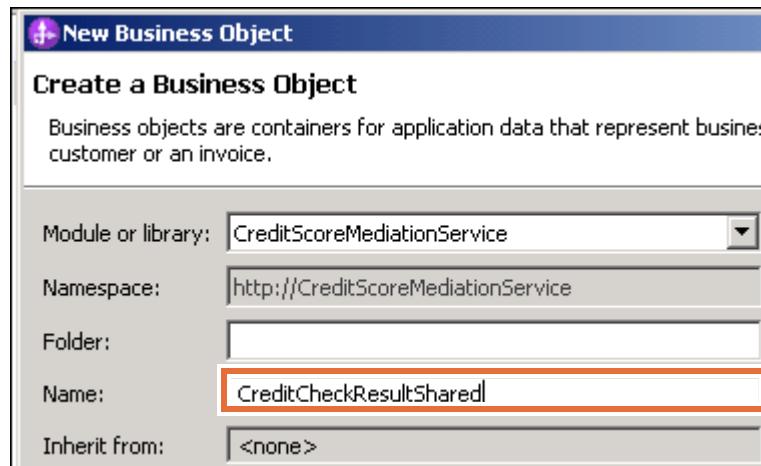
You implement the logic for the `CreditScoreMediation` in this exercise.

## Part 2: *Creating the required business object for the aggregation mediation flow*

Before you implement the logic in the mediation flow that starts the credit score providers, you create an extra business object for the mediation. This business object, `CreditCheckResultShared`, is a structure that contains the results that are returned from each of the credit score provider service invocations. This business object is stored in the *shared* context. Recall that the shared context is a thread-based location in memory that is shared across all instances of a service message object within a request flow or response flow. The shared context is used during an aggregation (fan-out, fan-in) mediation flow to temporarily store responses from service invocations.

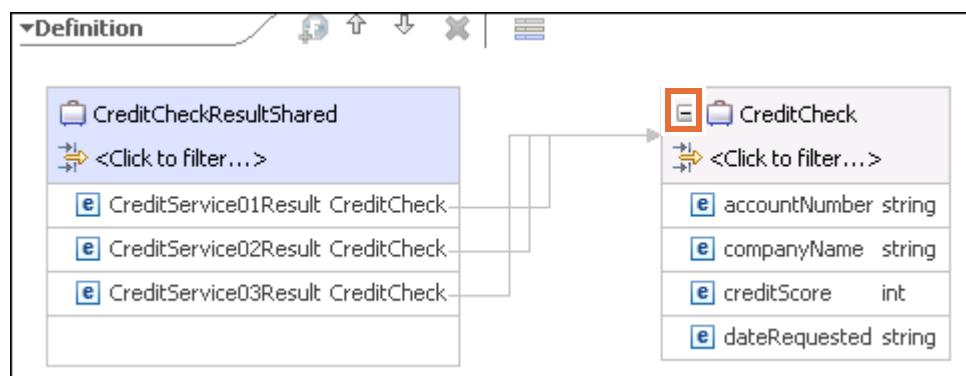
- \_\_\_ 1. Create new data type: `CreditCheckResultShared`
  - \_\_\_ a. In the **Business Integration** view, expand `CreditScoreMediationService`, right-click **Data**, and select **New > Business Object** from the menu.

- \_\_\_ b. In the **Name** field, enter `CreditCheckResultShared` and click **Finish**. The business object editor opens.



- \_\_\_ c. Right-click inside the business object, and choose **Add field** from the menu. You can also click the **Add a field to a business object** icon, the first icon in the toolbar to the right of the **Definition** section header.
- \_\_\_ d. Change the default name from `field1` to: `CreditService01Result`
- \_\_\_ e. Change the data type by clicking **string** and selecting `CreditCheck` from the list.
- \_\_\_ f. Right-click inside the business object, and choose **Add field** from the menu.
- \_\_\_ g. Change the default name to: `CreditService02Result`
- \_\_\_ h. Change the data type by clicking **string** and selecting `CreditCheck` from the list.
- \_\_\_ i. Right-click inside the business object, and choose **Add field** from the menu.
- \_\_\_ j. Change the default name to: `CreditService03Result`
- \_\_\_ k. Change the data type by clicking **string** and selecting `CreditCheck` from the list.

You defined the three elements of the business object. Optionally you can click the plus sign in front of the **CreditCheck** business object to see the fields that are defined inside it.



- \_\_\_ 2. Save the workspace (**Ctrl+S**). The newly created business object now shows in the **Data** hierarchy under the `CreditScoreMediationService` project.
- \_\_\_ 3. Close the business object editor pane.

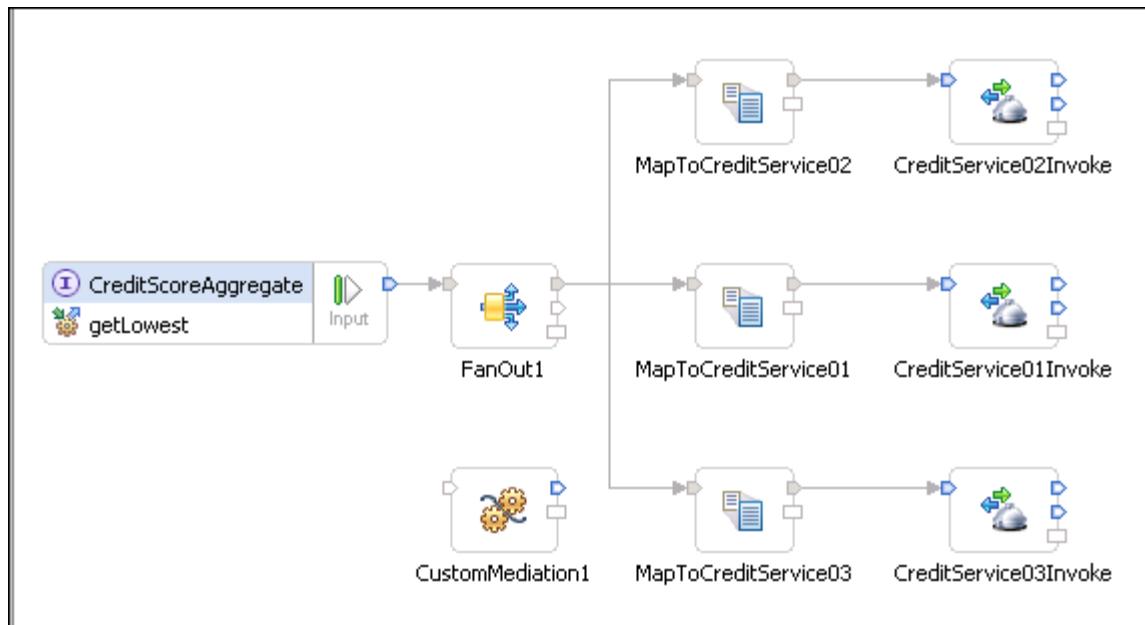
## Part 3: Implementing the mediation flow

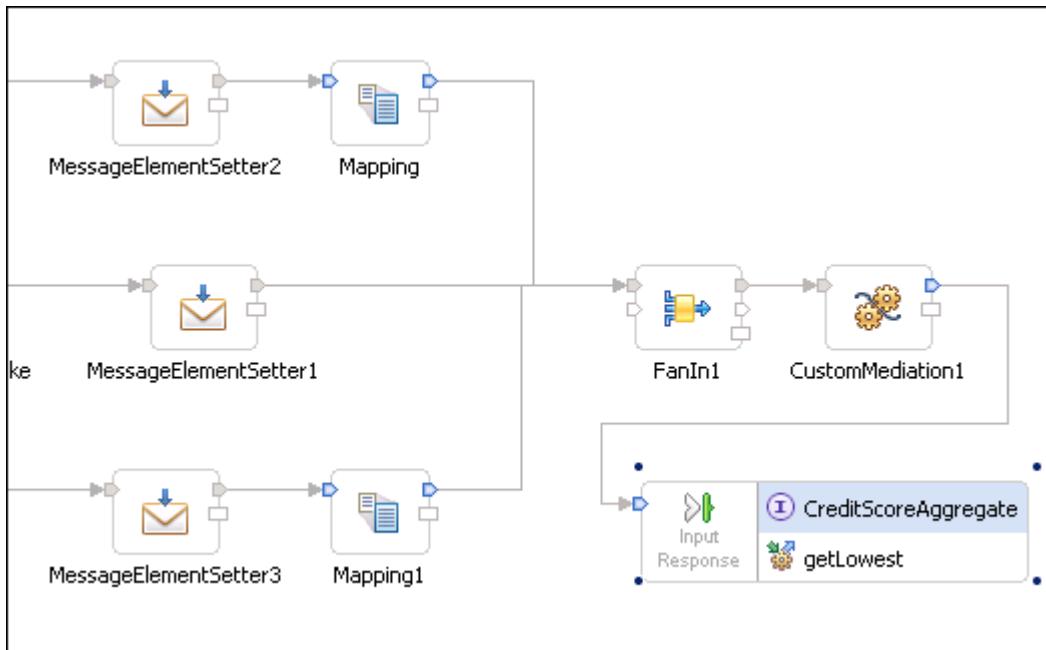
In this section, you implement the logic in the mediation module to start the three credit score providers. The logic also determines which of the returned credit scores is the lowest, and then returns that score to the invoking service.

To implement the mediation flow, you use the following primitives:

- A fan-out to start the aggregation. From this fan-out, three paths exist, one for each of the credit score service imports.
- A service invoke for each of the credit score services.
- An XSL transformation before each of the credit score invocations to convert the inbound message to the data type that each service invocation requires.
- A message element setter, which follows each of the credit score invocations to copy the response messages from the credit score services into the shared context. The custom mediation primitive queries the shared context.
- Two more XSL transformations, which follow the message element setter primitives for the second and third service invocations (described in further detail in the exercise).
- A fan-in to mark the end of the aggregation block.
- A custom mediation primitive that determines which of the three credit scores that are returned is the lowest. This primitive constructs the output message that is based on the response message that returned the lowest credit score.

The following diagram provides an overview of how the drawing canvas is shown when you finish constructing the mediation flow component. Each half of the diagram is shown separately for legibility.





Although the actual layout of your final diagram might be different, the wiring must be the same. You might want to look at this diagram as you are constructing the mediation flow component.



### Note

If you wire the primitives in a different order than is described in the instructions, some of the primitive names might not correspond to the diagrams. If it happens, do not be concerned. It does not affect the outcome of the exercise.

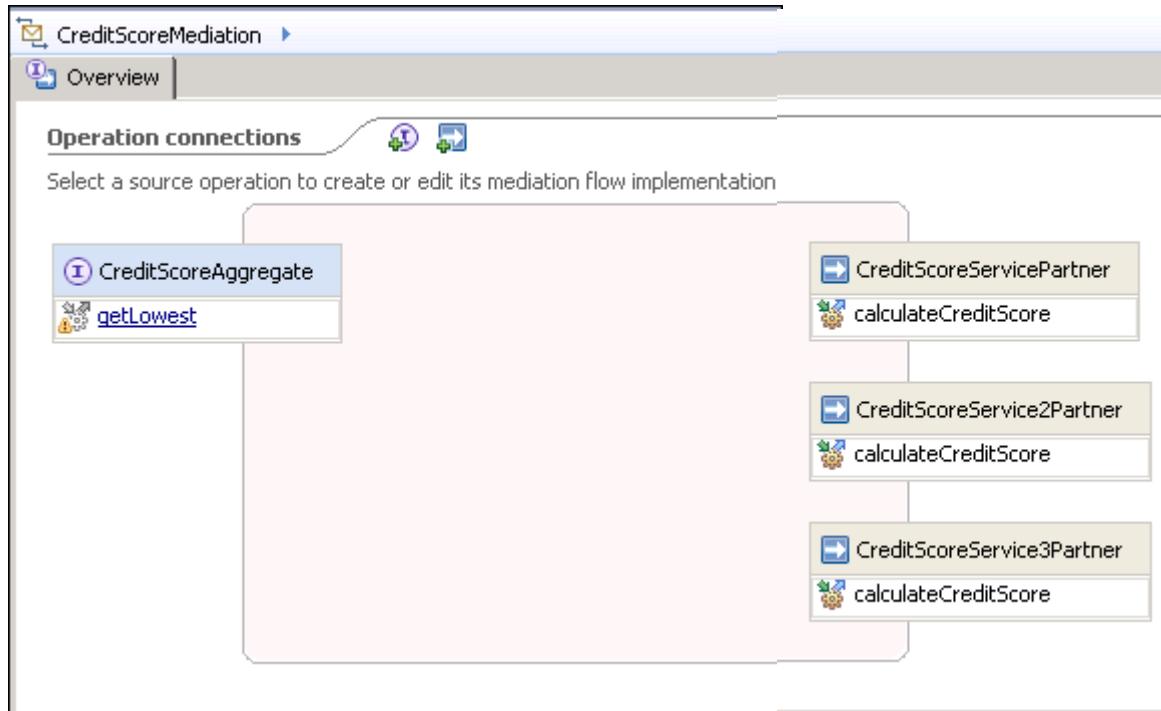
- 1. Review the mediation flow component.

- a. Switch to the `CreditScoreMediationService` assembly diagram.



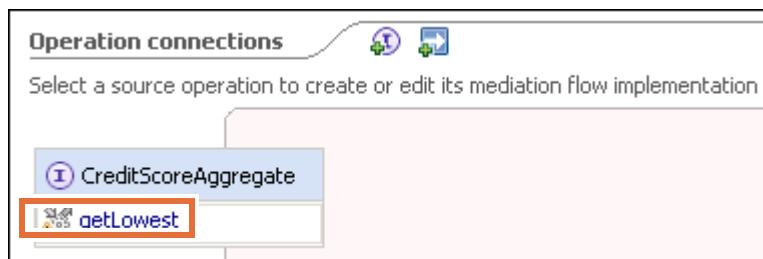
- b. Double-click the **CreditScoreMediation** mediation flow component.
  - c. In the **Tip** window, select the **Do not show tips** check box, and close the window.

- d. The **Operation connections** view of the component opens.



The interface and operation are shown on the left, while the credit services to start are on the right. As you construct the mediation flow, the Operation connections view is updated to reflect the invocation pattern.

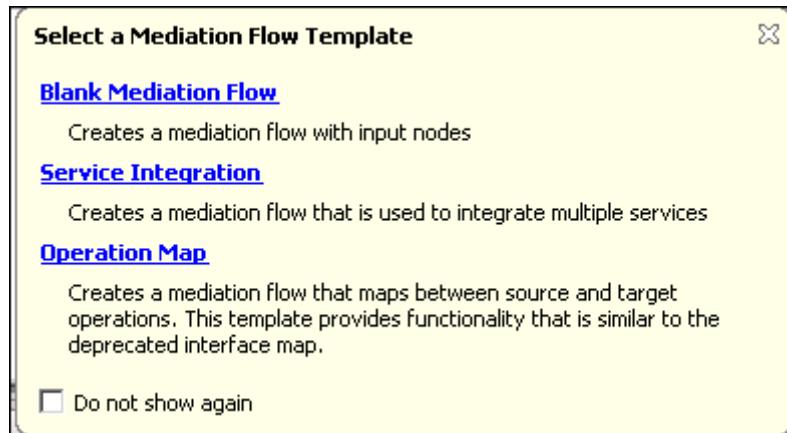
- 2. Open the request flow.
- a. In the **Operation connections** window, click the `getLowest` operation.



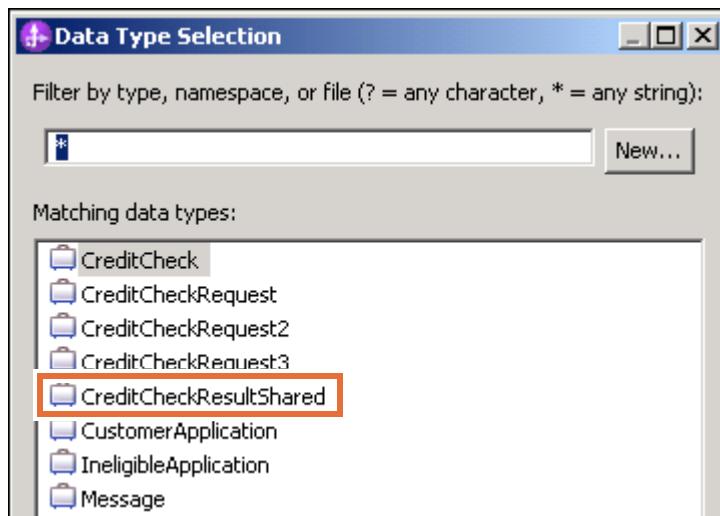
The mediation flow editor opens. The canvas contains an input node, an input response node, and a custom mediation primitive.

**Note**

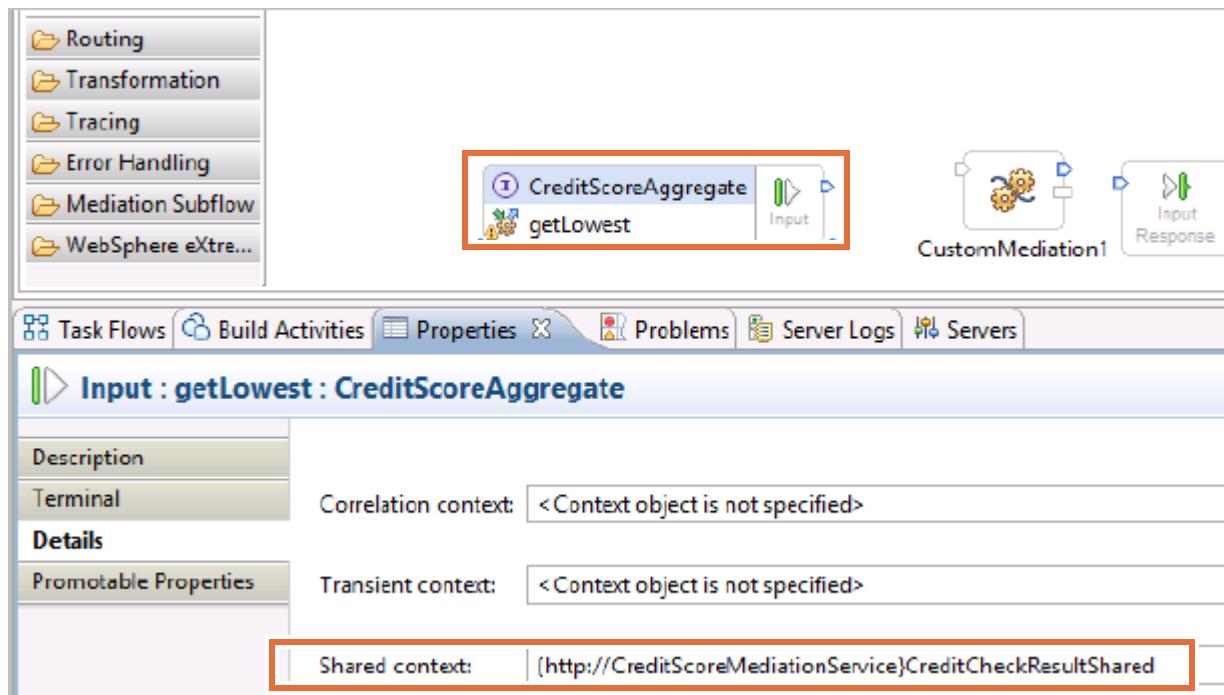
The custom mediation primitive for the mediation flow is provided for you (and the mediation flow editor for this mediation flow component was started previously). If these two conditions were not already set in place, you would see the mediation flow template wizard. From the wizard, you would select the option that you want to allow IBM Integration Designer to create an appropriate skeleton mediation flow.



- \_\_\_ 3. Configure the shared context.
  - \_\_\_ a. Right-click the **Input** node, and choose **Show In > Properties View** from the menu.
  - \_\_\_ b. Select the **Details** tab. The context attributes are shown.
  - \_\_\_ c. Click **Browse** to the right of the **Shared context** field. The **Data Type Selection** window opens.
  - \_\_\_ d. Select the `CreditCheckResultShared` data type, which you created previously.



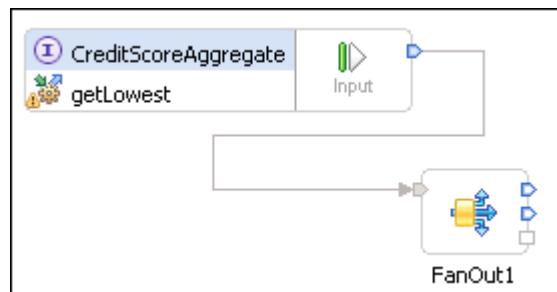
- \_\_ e. Click **OK**. The **Shared context** field is updated. You are now able to access the shared context with this data type.



### Note

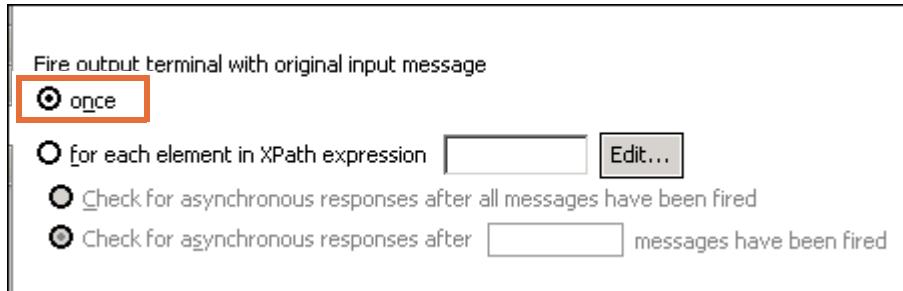
The correlation context and transient context are not used in this exercise.

- 
- \_\_ 4. Add and configure the fan-out primitive.
- \_\_ a. In the **Palette**, expand the **Routing** drawer, click **Fan Out**, and click the canvas to add the mediation primitive.
  - \_\_ b. Accept the default name.
  - \_\_ c. Wire the **out** terminal of the **Input** node to the **in** terminal of the **FanOut1** primitive.



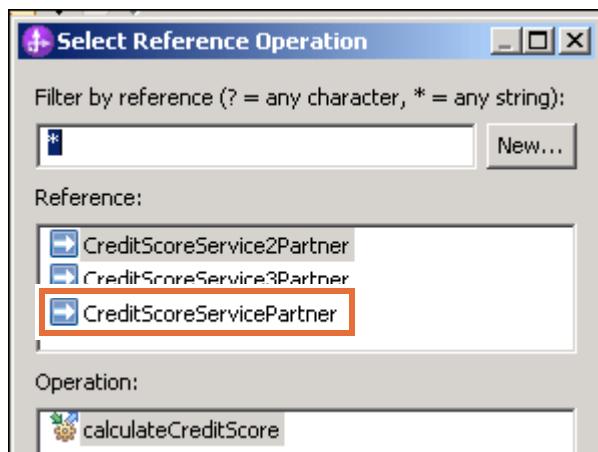
- \_\_ d. With the fan-out primitive selected, switch to the **Properties** view.
- \_\_ e. Click the **Details** tab.

- \_\_ f. Verify that the **Fire output terminal with original message** option is set to **once**.



In this mode (“default” mode), each message sent to the fan-out primitive is sent to each of the wires that are connected to its **out** terminal. That is, each wire that is connected to the **out** terminal receives an identical copy of the input message, in sequence. This message is received after the preceding branch was run if the decision point specified in the fan-in primitive was not yet reached.

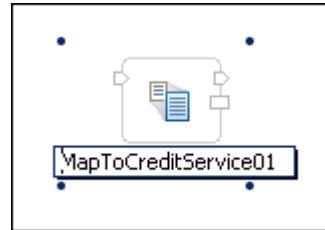
- \_\_ 5. Add and configure the service invoke primitives and related transformation primitives.
- \_\_ a. In the **Palette**, expand the **Service Invocation** drawer, click **Service Invoke**, and click the canvas to add the mediation primitive to start the first credit score service. The **Select Reference Operation** window opens.
  - \_\_ b. Select **CreditScoreServicePartner** as the reference.



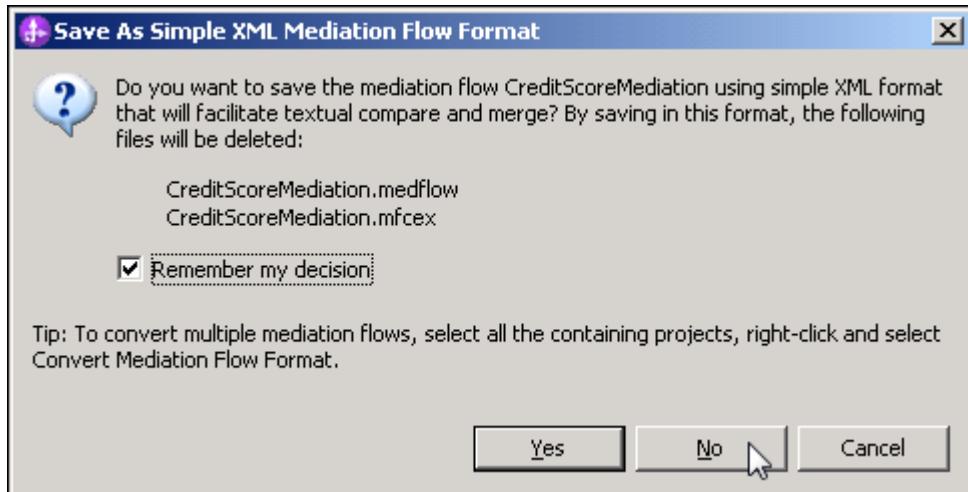
- \_\_ c. Click **OK**. The primitive is placed on the canvas. Position it to the right of the fan-out primitive.
- \_\_ d. In the **Properties** view, on the **Description** tab, change the **Display Name** of the primitive to: **CreditService01Invoke**
- \_\_ e. The output parameter of the **FanOut** primitive does not match the input parameter of the **Service Invoke** primitive. You must insert a data map transformation between the two primitives to transform the data types.

Open the **Transformation** tools in the palette, click **Mapping**, and click anywhere in the workspace to add the transformation primitive to the editor.

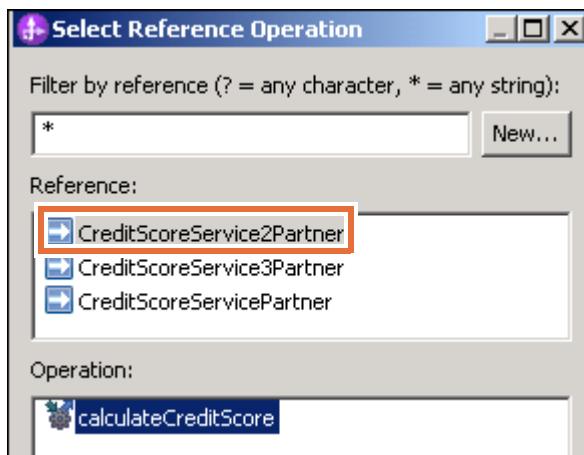
- \_\_ f. Set the **Display name** of the transformation primitive to: **MapToCreditService01**



- \_\_ g. Wire the **out** terminal of the **FanOut1** primitive to the **in** terminal of **MapToCreditService01**.
- \_\_ h. Wire the **out** terminal of the transformation map to the **in** terminal of **CreditService01Invoke**.
- \_\_ i. Save the workspace. If you are prompted to save the workspace in Simple XML format, check the **Remember my decision** check box and click **No**.



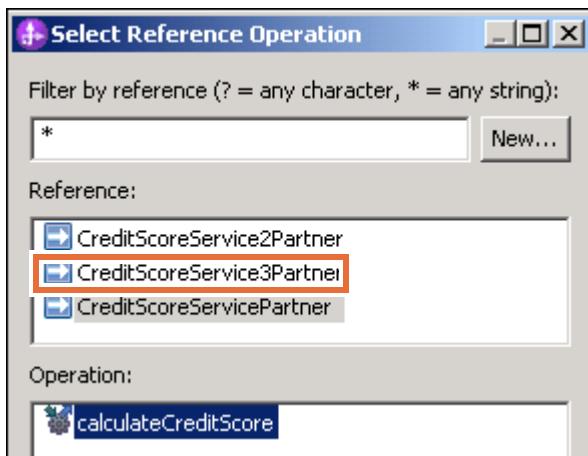
- \_\_ j. Add another **Mapping** primitive named: **MapToCreditService02**
- \_\_ k. Add another **Service Invoke** primitive for the second credit service. The primitive uses the **CreditScoreService2Partner** as the reference.



- \_\_ l. Click **OK**. The primitive is placed on the canvas. Position the primitive below the **CreditService01Invoke** service invoke primitive.

- \_\_ m. On the **Description** tab in the **Properties** view, change the **Display Name** to: CreditService02Invoke
- \_\_ n. Wire the **out** terminal of the fan-out primitive to the **in** terminal of the MapToCreditService02 transformation primitive.
- \_\_ o. Wire the **out** terminal of the transformation primitive to the **in** terminal of the CreditService02Invoke service invoke.
- \_\_ p. Add a third **Mapping** primitive to the flow. Name the primitive: MapToCreditService03
- \_\_ q. Add a third **Service Invoke** primitive to the flow. Name the primitive: CreditService03Invoke

This invocation primitive uses the **CreditScoreService3Partner** as the reference.

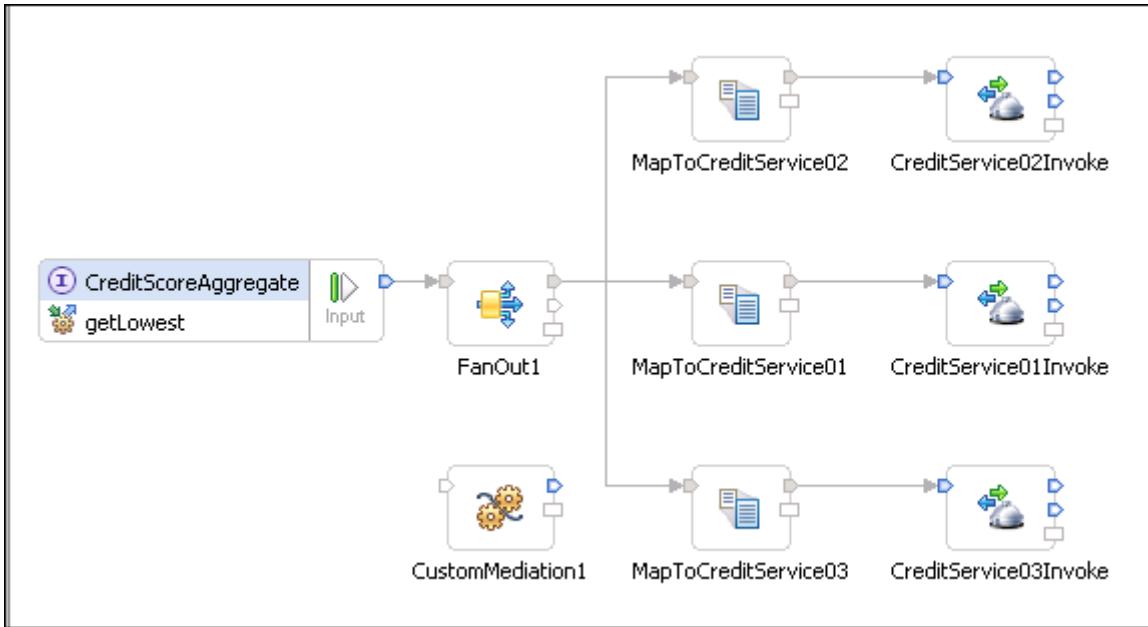


- \_\_ r. Click **OK**. The primitive is placed on the canvas. Position the primitive below the CreditService02Invoke service invoke primitive.
- \_\_ s. Wire the **out** terminal of the fan-out primitive to the **in** terminal of MapToCreditService03.
- \_\_ t. Wire the **out** terminal of MapToCreditService03 to the **in** terminal of CreditService03Invoke.
- \_\_ u. Save your workspace. IBM Integration Designer generates errors on the transformation primitives because you did not yet implement them. Ignore the errors. Your diagram resembles the one that follows:



### Hint

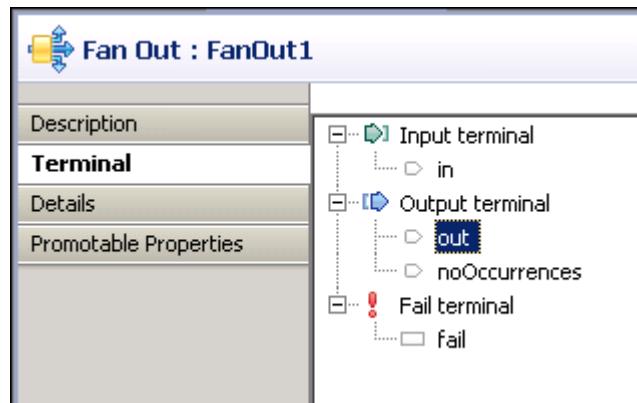
Right-click anywhere on the white space of the assembly diagram and select **Layout contents** to rearrange the diagram and get better view of the component wiring.



### Note

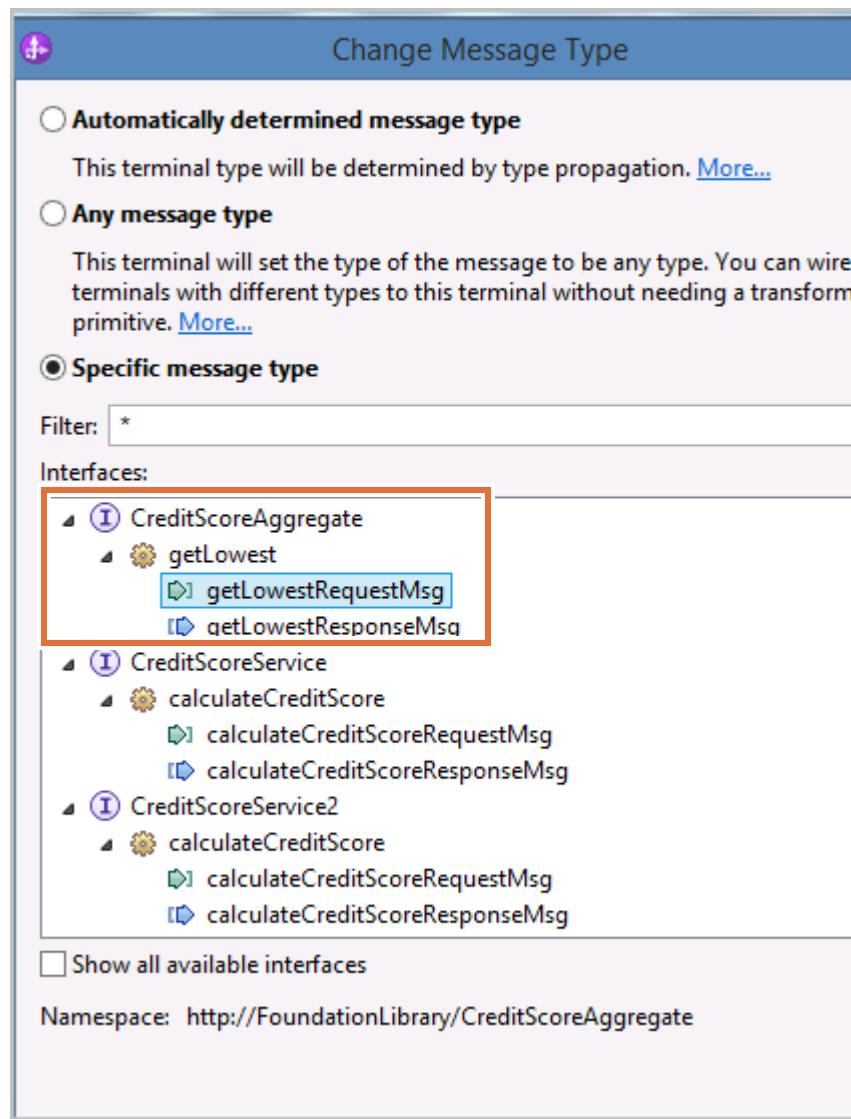
The order and layout of primitives from top to bottom might be different in your case. You are okay if you wired the right primitives according to the instructions.

- \_\_\_ 6. Implement the data map transformation primitives.
  - \_\_\_ a. Select the **FanOut1** primitive.
  - \_\_\_ b. Select the **Properties** view and click the **Terminal** tab. Click the **out** terminal beneath the **Output terminal** section.



- \_\_\_ c. Click **Change**, which is next to the **Message type** field.

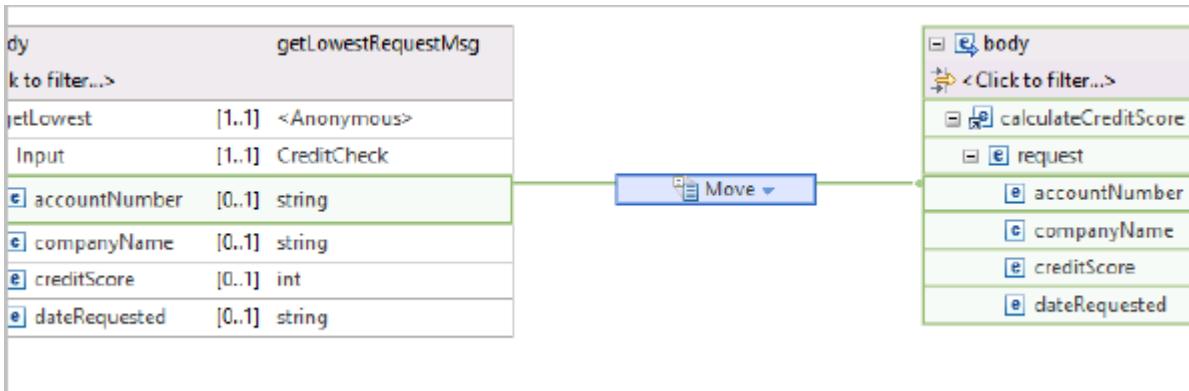
- \_\_\_ d. In the **Change Message Type** window, click **Specific message type**. In the **Interfaces** section, click **CreditScoreAggregate > getLowest > getLowestRequestMsg**.



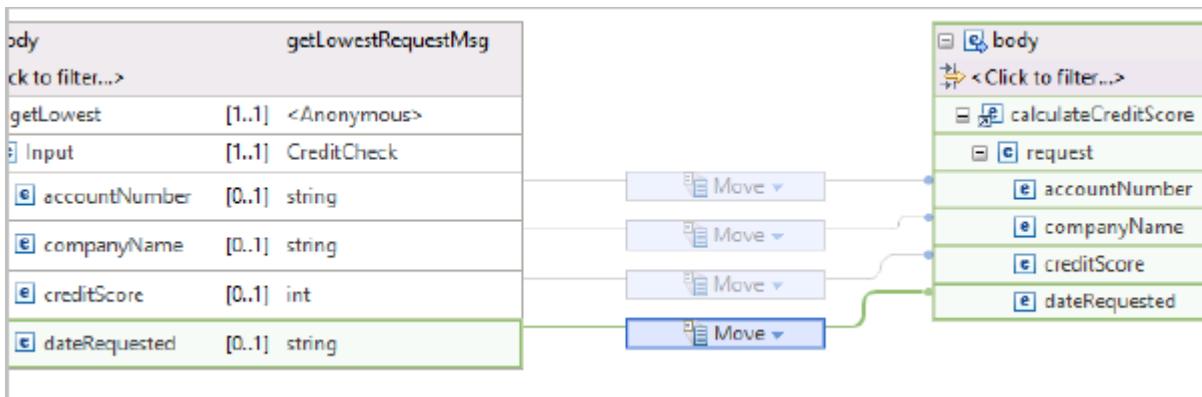
Click **OK**.

- \_\_\_ e. Select the **MapToCreditService01** transformation primitive and click the **Properties** view. Click the **Details** tab.
- \_\_\_ f. The **Mapping file** field is not presently set. Click **New**.
- \_\_\_ g. The **New XML Map** window opens. Accept the default values and click **Finish**. The mapping editor opens.
- \_\_\_ h. Expand all the levels of the **getLowest** operation of **getLowestRequestMsg** and all the levels of the **calculateCreditScore** operation of **calaculateCreditScoreRequestMsg**.

- \_\_ i. Drag `accountNumber` from the **Input: CreditCheck** parameter of `getLowestRequestMsg` to the field of the same name in **request: CreditCheckRequest**. A **Move** transformation is created.



- \_\_ j. Similarly, move the remaining fields from `getLowestRequestMsg` to `calculateCreditScoreRequestMsg`.

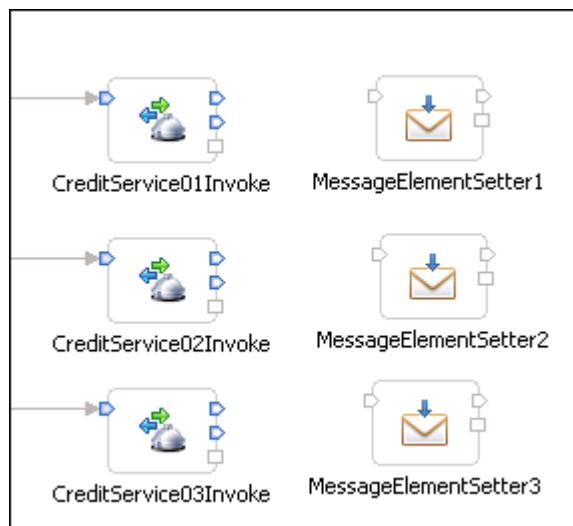


### Hint

The subordinate fields are identically named and typed. You can also use the automap feature to automatically add **Move** transformations.

- 
- \_\_ k. Save the map and close the mapping editor.
- \_\_ l. Select the `MapToCreditService02` transformation primitive and click the **Properties** view. Click the **Details** tab.
- \_\_ m. The **Mapping file** field is not presently set. Click **New**.
- \_\_ n. The **New XML Map** window opens. Accept the default values and click **Finish**. The mapping editor opens.
- \_\_ o. Expand all the levels of the `getLowest` operation of `getLowestRequestMsg` and all the levels of the `calculateCreditScore` operation of `calaculateCreditScoreRequestMsg`.
- \_\_ p. Create **Move** transformations to move the fields from the **Input: CreditCheck** parameter to the fields of the same name in **request: CreditCheckRequest2**.

- \_\_\_ q. Save the map and close the mapping editor.
  - \_\_\_ r. Select the **MapToCreditService03** transformation primitive and click the **Properties** view. Click the **Details** tab.
  - \_\_\_ s. The **Mapping file** field is not presently set. Click **New**.
  - \_\_\_ t. The **New XML Map** window opens. Accept the default values and click **Finish**. The mapping editor opens.
  - \_\_\_ u. Expand all the levels of the **getLowest** operation of **getLowestRequestMsg** and all the levels of the **calculateCreditScore** operation of **calaculateCreditScoreRequestMsg**.
  - \_\_\_ v. Create **Move** transformations to move the fields from the **Input: CreditCheck** parameter to the fields of the same name in **request: CreditCheckRequest3**.
  - \_\_\_ w. Save the map and close the mapping editor.
  - \_\_\_ x. Save the mediation flow. The Problems view should have no errors.
- 7. Add and configure the message element setter primitives that store the response messages from the service invocations into the shared context. Each credit service invoke returns its response in the **calculateCreditScoreReturn** element of the body of the service message object. For each of the three service invocations, you store the corresponding response messages in a specific location in the **CreditCheckResultShared** shared context (which you defined earlier in the exercise).
- \_\_\_ a. From the **Transformation** drawer, add three **Message Element Setter** mediation primitives to the canvas. Position each mediation primitive to the right of the three service invoke primitives. The primitives must be placed in an arrangement such that the first, **MessageElementSetter1** mediation primitives, are placed next to the **CreditServiceInvoke1** service invoke primitive. The second, **MessageElementSetter2** mediation primitives, must be placed next to the **CreditServiceInvoke2** service invoke primitive. The third, **MessageElementSetter3** mediation primitives, must be placed next to the **CreditServiceInvoke3** service invoke primitive.

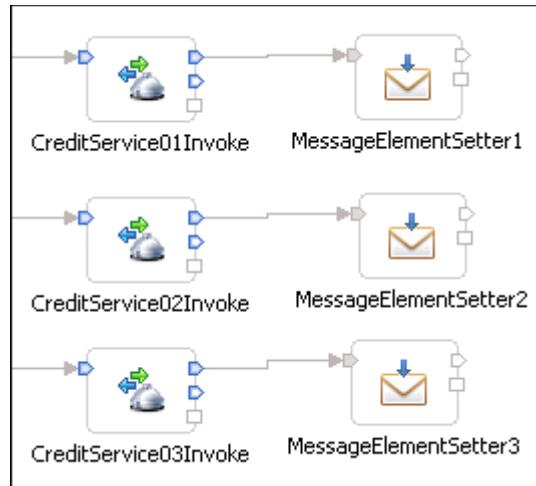




## Note

The order and layout of primitives from top to bottom might be different in your case.

- \_\_ b. Wire the **out** terminal of each of the service invoke primitives to the **in** terminal of the corresponding message element setter primitives.



- \_\_ c. Open the properties of **MessageElementSetter1**. Select the **Details** tab. To the right of the **Message Elements** matrix, click **Add**.

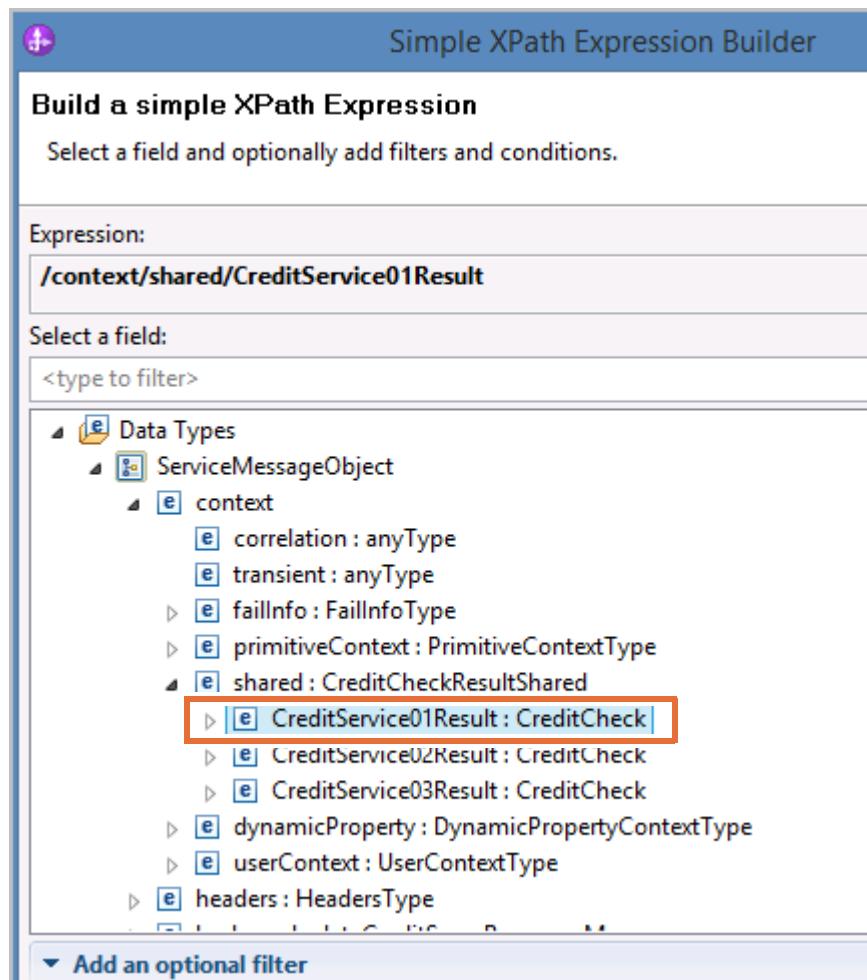
**Message Element Setter : MessageElementSetter1**

|                       |                                                                                                                    |  |  |
|-----------------------|--------------------------------------------------------------------------------------------------------------------|--|--|
| Description           | Message Elements:                                                                                                  |  |  |
| Terminal              |                                                                                                                    |  |  |
| <b>Details</b>        | <input type="button" value="Add..."/> <input type="button" value="Edit..."/> <input type="button" value="Remove"/> |  |  |
| Promotable Properties |                                                                                                                    |  |  |

The **Add/Edit** window opens.

- \_\_ d. Set the **Action** field to **Copy**.
  - \_\_ e. To the right of the **Target** field, click **Browse**. The XPath expression builder window opens.

Browse through the **ServiceMessageObject** to the location /context/shared/CreditService01Result, and then click **OK**.



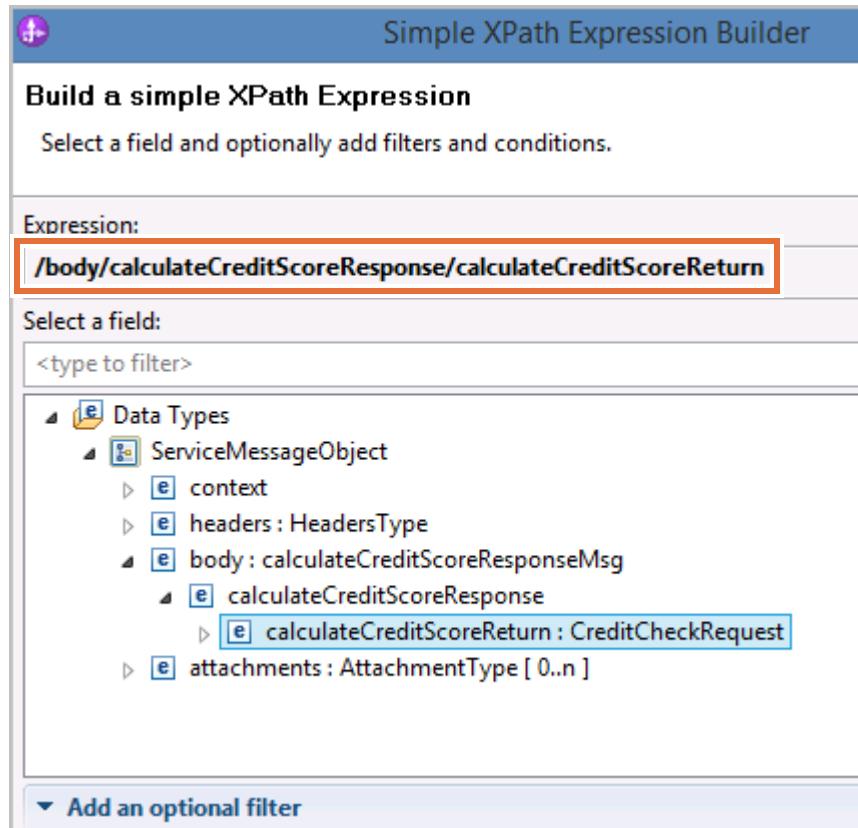
The **Target** field is populated.



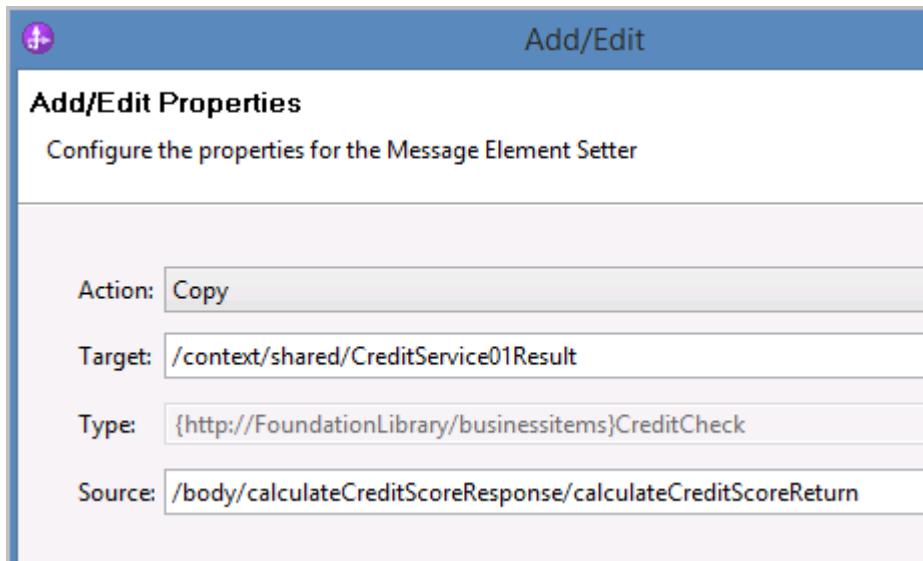
### Note

If you know the location that you want to enter, you can type it directly into the **Target** field without using the XPath expression builder. The content assist feature helps you complete the typing in the field.

- \_\_ f. Enter the value in the **Source** field, either by typing in the XPath expression `/body/calculateCreditScoreResponse/calculateCreditScoreReturn` or by browsing through the SMO to that location, and then click **OK**.



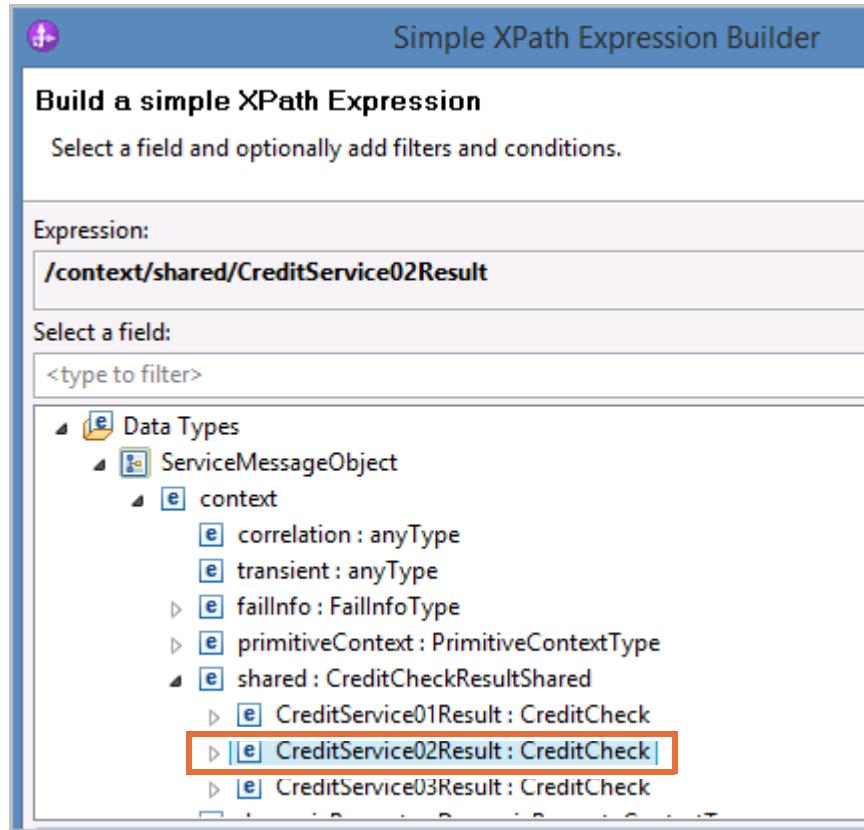
- \_\_ g. The **Source** field is populated and the **Add/Edit** window is complete.



- \_\_ h. Click **Finish**. The **Message Elements** matrix is updated. If you need to modify an existing message element, select it and then click **Edit**.

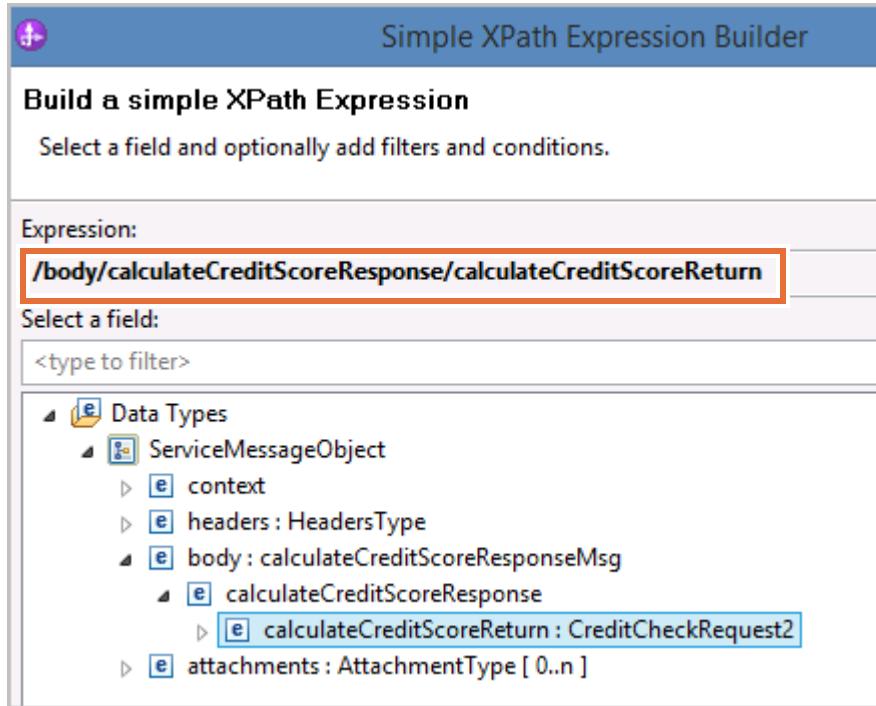
Now you repeat the configuration steps for the other two message element setter primitives.

- \_\_ i. Open the properties of **MessageElementSetter2**. Select the **Details** tab. To the right of the **Message Elements** matrix, click **Add**. The **Add/Edit** window opens.
- \_\_ j. Set the **Action** field to **Copy**.
- \_\_ k. To the right of the **Target** field, click **Browse**. The XPath expression builder window opens.
- \_\_ l. Browse through the **ServiceMessageObject** to the location `/context/shared/CreditService02Result`, and then click **OK**.

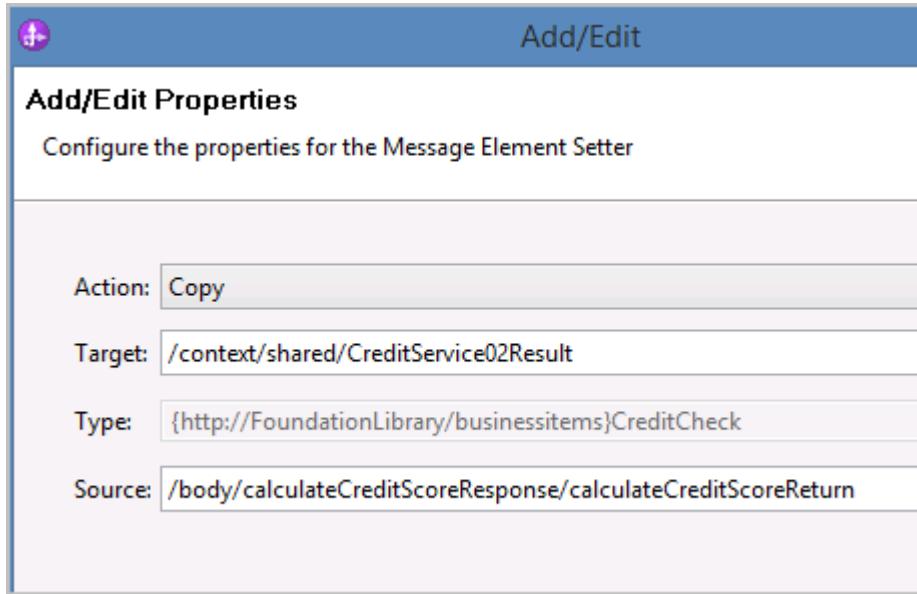


The **Target** field is populated.

- \_\_\_ m. Enter the value in the **Source** field, either by typing in the XPath expression `/body/calculateCreditScoreResponse/calculateCreditScoreReturn` or by browsing through the SMO to that location. Click **OK**.

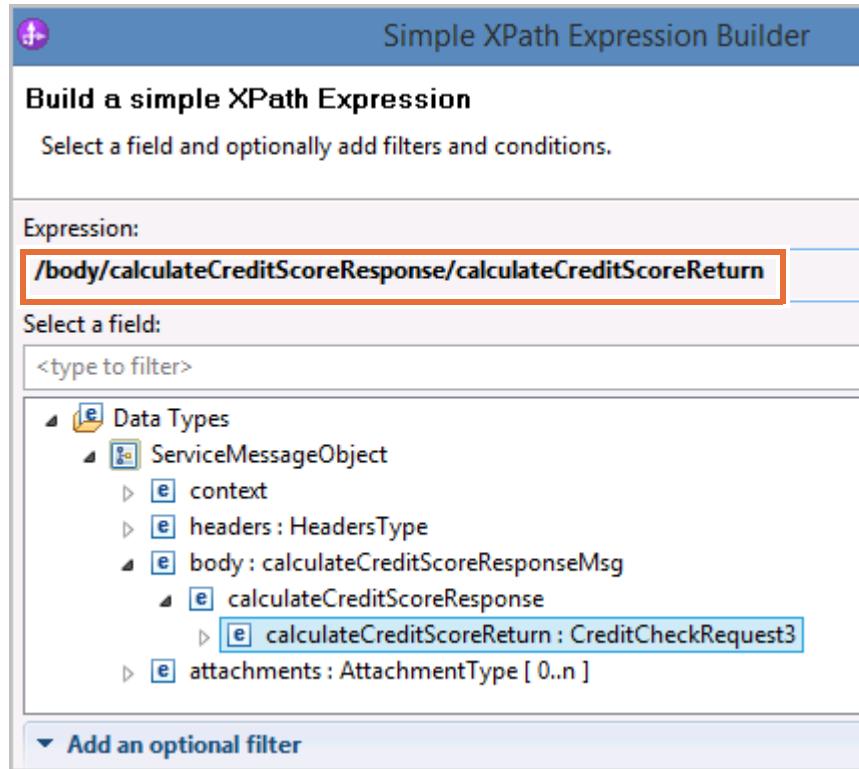


- \_\_\_ n. The **Source** field is populated and the **Add/Edit** window is complete.



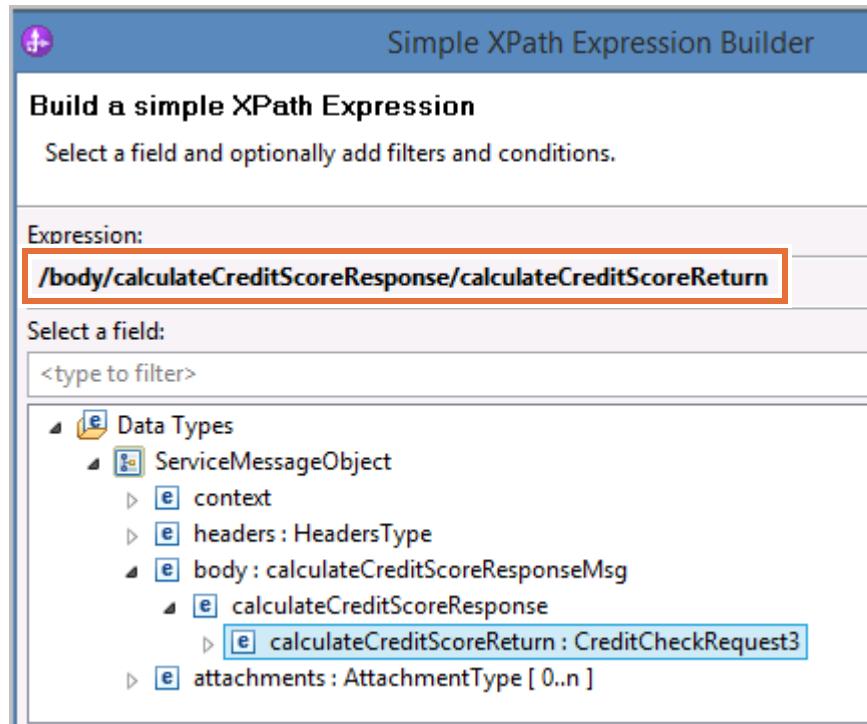
- \_\_\_ o. Click **Finish**. The **Message Elements** matrix is updated.  
 \_\_\_ p. Open the properties of **MessageElementSetter3**. Select the **Details** tab. To the right of the **Message Elements** matrix, click **Add**. The **Add/Edit** window opens.  
 \_\_\_ q. Set the **Action** field to **Copy**.

- \_\_\_ r. To the right of the **Target** field, click **Browse**. The XPath expression builder window opens.
- \_\_\_ s. Browse through the **ServiceMessageObject** to the location /context/shared/CreditService03Result, and then click **OK**.

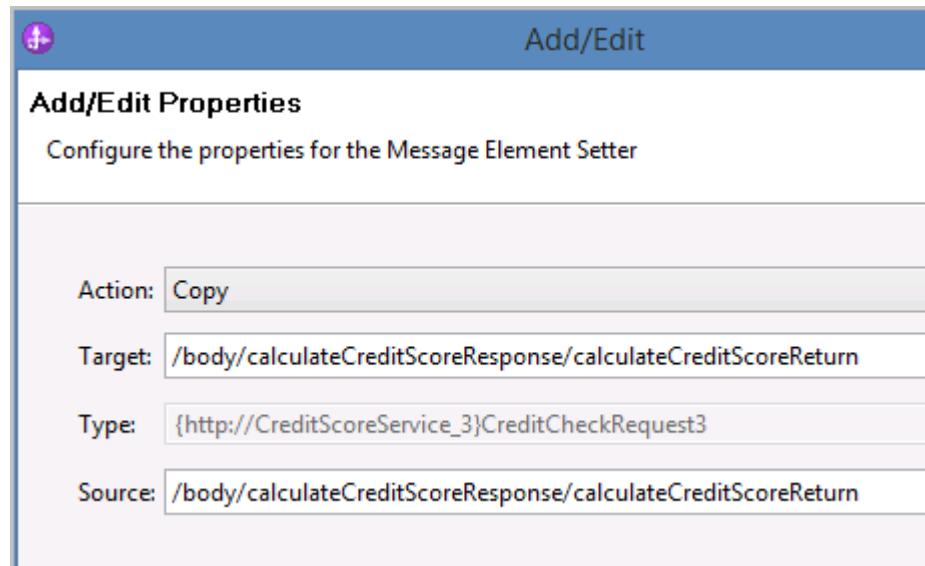


The **Target** field is populated.

- \_\_ t. Enter the value in the **Source** field, either by typing in the XPath expression `/body/calculateCreditScoreResponse/calculateCreditScoreReturn` or by browsing through the SMO to that location. Click **OK**.

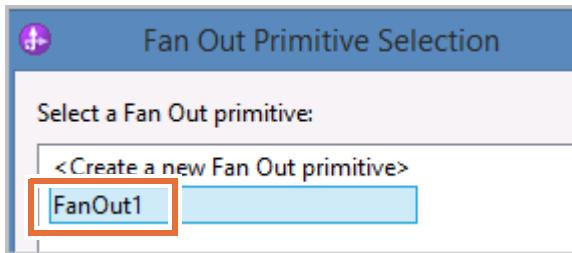


- \_\_ u. The **Source** field is populated and the **Add/Edit** window is complete.



- \_\_ v. Click **Finish**. The **Message Elements** matrix is updated.  
\_\_ 8. Save the workspace (**Ctrl+S**).

- \_\_\_ 9. Add and configure the fan-in primitive and related data transformations.
  - \_\_\_ a. From the **Routing** palette, select the **Fan In** mediation primitive. Place it on the canvas to the right of **MessageElementSetter2**. The **Fan Out Primitive Selection** window opens. Recall that a fan-in primitive must always be paired with a fan-out, but you can use a fan-out primitive alone.
  - \_\_\_ b. Select **FanOut1**, and then click **OK**.



- \_\_\_ c. View the properties of the fan-in primitive. In the **Details** tab, select the option **Fire output terminal when \_\_\_\_\_ input messages have been received**, and set the value to 3.



In this mode, the fan-in waits for the three messages to arrive through the **in** terminal, and then it fires its **out** terminal.

- \_\_\_ d. Wire the **out** terminal of **MessageElementSetter1** to the **in** terminal of the fan-in primitive.

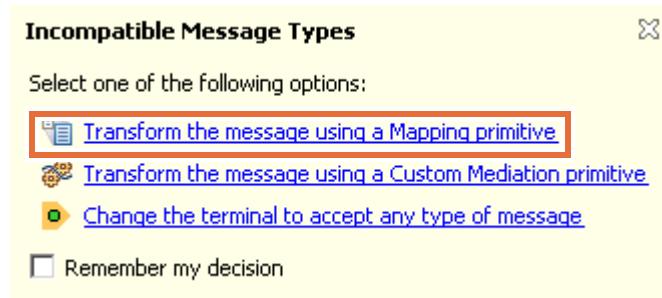


### Note

Be sure to wire to the **in** terminal of the fan-in primitive, not the **stop** terminal.

- \_\_\_ e. Wire the **out** terminal of **MessageElementSetter2** to the **in** terminal of the fan-in primitive. When you wire these terminals, IBM Integration Designer alerts you to a data type mismatch and presents a wizard to help resolve it.

- \_\_\_ f. Select the **Transform the message using a Mapping primitive** option. A mapping primitive is inserted between the message element setter primitive and the fan-in primitive.



- \_\_\_ g. Double-click the mapping transformation primitive, which is inserted. The **New XML Map** window opens.
- \_\_\_ h. Click **Finish**. The mapping editor opens.
- \_\_\_ i. Expand the source and target business objects.
- \_\_\_ j. Create **move** transformations between the four elements of the **calculateCreditScoreReturn** structure of the source and target business elements:
- Map accountNumber to accountNumber
  - Map companyName to companyName
  - Map creditScore to creditScore
  - Map dateRequested to dateRequested



### Hint

You can also use the automap feature.

- \_\_\_ k. Save the XML map (**Ctrl+S**).
- \_\_\_ l. Close the mapping editor.



### Questions

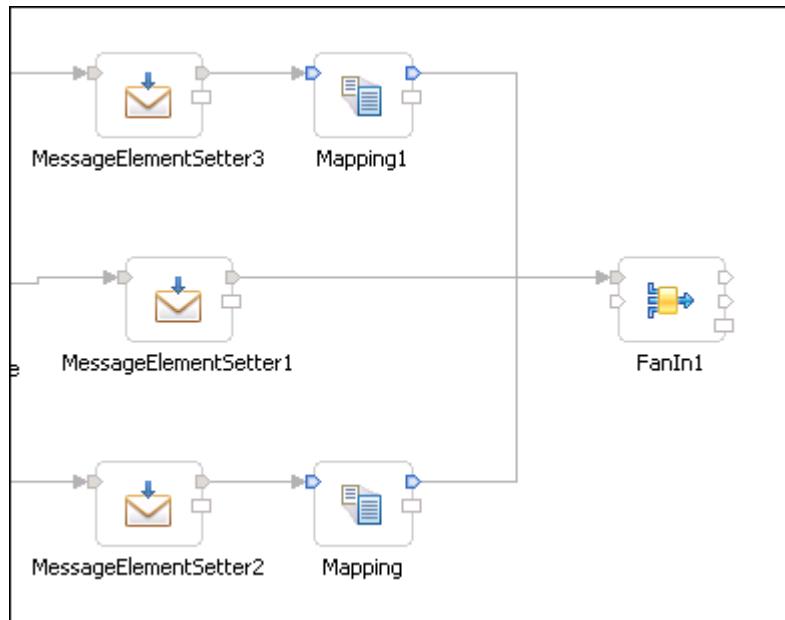
Why does wiring the second message element setter to the fan-in cause a type mismatch, but wiring the first message element setter did not?

When the fan-in primitive was originally added to the canvas, the **in** terminal of the primitive was untyped. Wiring the first message element setter primitive to the fan-in primitive established the message type of the **in** terminal. When you attempted to wire the second message element setter primitive to the fan-in, a type mismatch was detected.

- \_\_\_ m. Wire the **out** terminal of **MessageElementSetter3** to the **in** terminal of the fan-in primitive. Again, IBM Integration Designer alerts you to a data type mismatch and presents a wizard to help resolve it.

- \_\_ n. Select the **Mapping primitive** option. A Mapping primitive is inserted between the message element setter primitive and the fan-in primitive.
- \_\_ o. Double-click the mapping primitive. The **New XML Map** window opens.
- \_\_ p. Click **Finish**. The mapping editor opens.
- \_\_ q. Expand the source and target business objects.
- \_\_ r. Create **move** transforms between the four elements of the **calculateCreditScoreReturn** structure of the source and target business elements:
  - Map accountNumber to accountNumber
  - Map companyName to companyName
  - Map creditScore to creditScore
  - Map dateRequested to dateRequested
- \_\_ s. Save the XML map (**Ctrl+S**).
- \_\_ t. Close the mapping editor.

The message element setter primitives, the fan-in primitive, and the associated XSL transformation primitives are now complete. The names of your transformation primitives might be different from those primitives depicted.

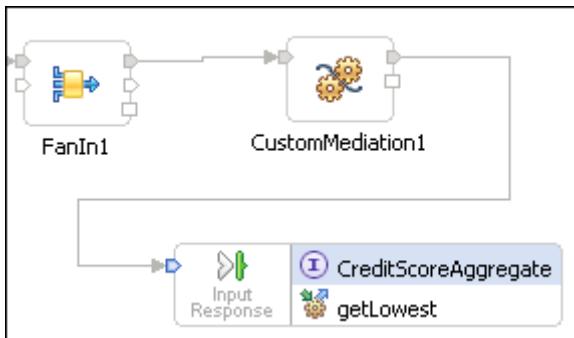


- \_\_ 10. Save the workspace (**Ctrl+S**).
  - \_\_ 11. Wire the custom mediation primitive and input response node into the mediation flow.
- The custom mediation primitive is supplied for you. It determines which of the responses (stored in the shared context) contains the lowest credit score; it then builds a response business object, which contains the response message from the corresponding service invoke.
- \_\_ a. Wire the **out** terminal of the fan-in primitive to the **in** terminal of the **CustomMediation1** primitive.

**Note**

Be sure to wire the **out** terminal of the fan-in primitive, not the **incomplete** terminal.

- \_\_\_ b. Wire the **out** terminal of the **CustomMediation1** primitive to the **in** terminal of the **Input Response** node.



- \_\_\_ 12. Save the workspace (**Ctrl+S**).
- \_\_\_ 13. Close the **CreditScoreMediation** flow editor pane.

#### **Part 4: Testing the solution**

You test the module with the integration test client.

- \_\_\_ 1. Start the server, unless it is already running.
  - \_\_\_ a. In the **Servers** tab, right-click the **IBM Process Server v8.5.7 at localhost** server.
  - \_\_\_ b. From the menu, click **Start**.
  - \_\_\_ c. Wait until the server starts. The server log shows the message **Server server1 open for e-business**, and the server state is **[Started, Synchronized]**.
- \_\_\_ 2. Deploy the application components.
  - \_\_\_ a. In the **Servers** view, right-click **IBM Process Server v8.5.7 at localhost**.
  - \_\_\_ b. Select **Add and Remove** from the menu.
  - \_\_\_ c. Click **Add All** to add them to the **Configured** list.
  - \_\_\_ d. Click **Finish**. The projects are deployed to the server. Wait for the deployment process to complete. When deployment is complete, the server status returns to **[Started, Synchronized]**. It might take several minutes. Additionally, in the **Server Logs** view, you see several messages, which indicate that the deployed applications started successfully.
- \_\_\_ 3. Test the **CreditScoreMediation** component.
  - \_\_\_ a. In the **CreditScoreMediationService** assembly diagram, right-click the **CreditScoreMediation** mediation flow component and select **Test Component** from the menu. The integration test client starts and shows the events and properties window.
  - \_\_\_ b. In the **Configuration** field, verify that **Default Module Test** is selected.

- \_\_ c. In the **Module** field, verify that **CreditScoreMediationService** is selected.
- \_\_ d. In the **Component** field, verify that **CreditScoreMediation** is selected.
- \_\_ e. In the **Interface** field, verify that **CreditScoreAggregate** is selected.
- \_\_ f. In the **Operation** field, verify that **getLowest** is selected.

The screenshot shows the 'General Properties' dialog with the 'Detailed Properties' section expanded. It contains a note about specifying component, interface, operation, and input parameter values for the Invoke event. Below this, a configuration section lists the following parameters:

|                       |                             |
|-----------------------|-----------------------------|
| <u>Configuration:</u> | Default Module Test         |
| <u>Module:</u>        | CreditScoreMediationService |
| <u>Component:</u>     | CreditScoreMediation        |
| <u>Interface:</u>     | CreditScoreAggregate        |
| <u>Operation:</u>     | getLowest                   |

- \_\_ g. In the **Initial request parameters** section, enter **IBM** in the **companyName** field.

The screenshot shows the 'Initial request parameters' editor. The 'Value editor' radio button is selected. A table lists the parameters:

| Name          | Type        | Val      |
|---------------|-------------|----------|
| Input         | CreditCheck | [ab]     |
| accountNumber | string      | [ab]     |
| companyName   | string      | [ab] IBM |
| creditScore   | int         | [ab] 0   |
| dateRequested | string      | [ab]     |

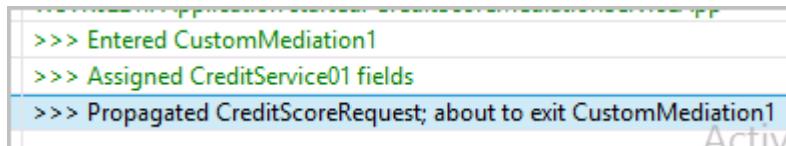
- \_\_ h. In the **Events** toolbar, click **Continue** (the first icon).

The screenshot shows the 'Events' toolbar. It includes a note about displaying events in a test trace and selecting an event to view its properties. The toolbar features several icons, with the 'Invoke' icon being the first one highlighted by a red box.

- \_\_ i. If you are prompted for Deployment Location, choose default **IBM Process Server v8.5.7 at localhost** and then click **Finish**.
- \_\_ j. If you are prompted, use **User Name** `admin` and **Password** `web1sphere` and then click **OK**.

The test client initiates the test with the supplied data. It might take several minutes.

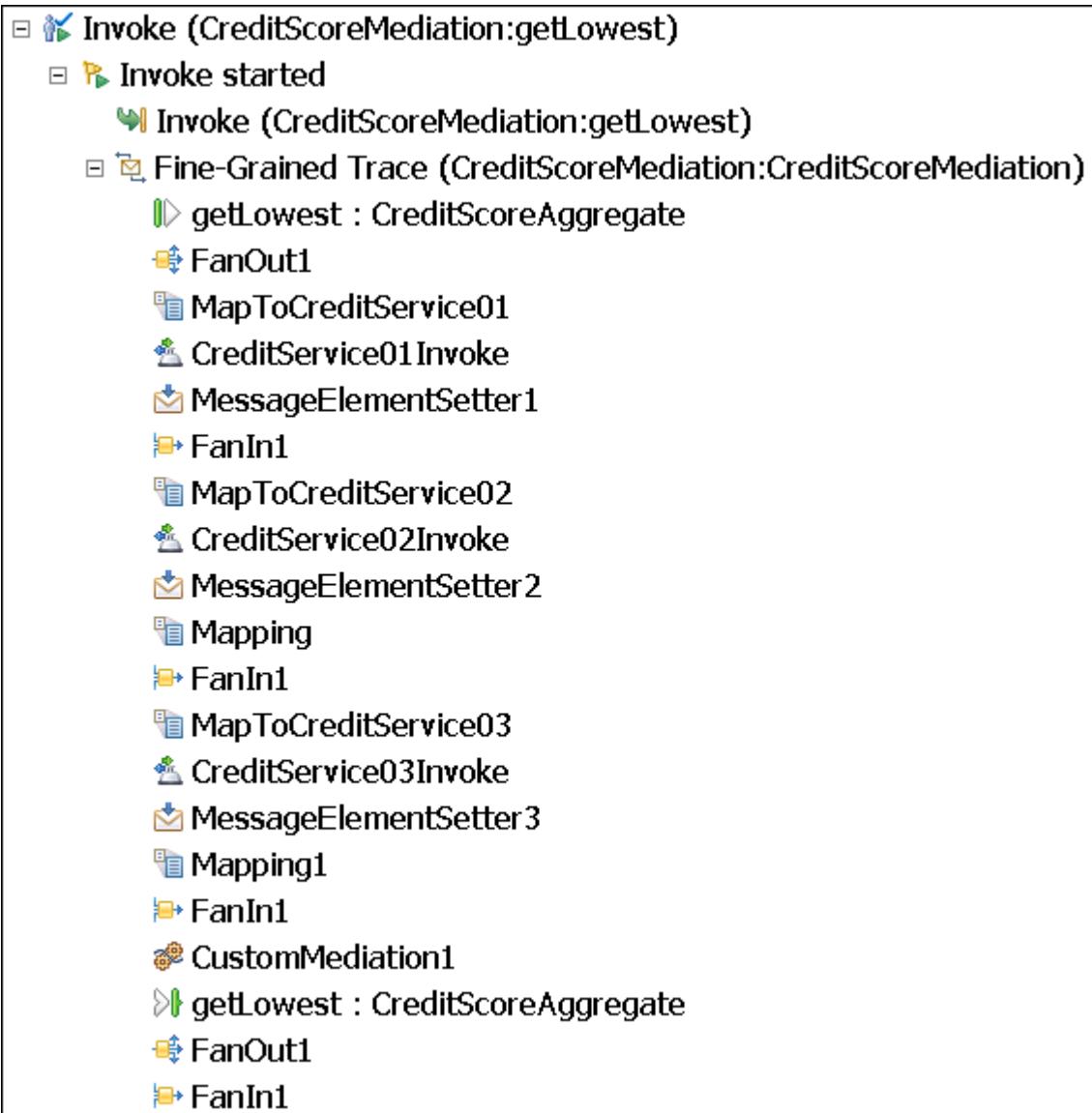
- \_\_ 4. Review the test results.
  - \_\_ a. When the test completes, switch to the **Server Logs** tab.
  - \_\_ b. Examine the server log for a series of log messages that the custom mediation primitive writes. They are shown at the bottom of the server log.



A screenshot of the IBM Process Server server logs interface. The logs are displayed in a table with three columns: timestamp, log message, and severity level (labeled 'Activ' in the screenshot). The log entries are:

|                                                                   |  |       |
|-------------------------------------------------------------------|--|-------|
| >>> Entered CustomMediation1                                      |  | Activ |
| >>> Assigned CreditService01 fields                               |  | Activ |
| >>> Propagated CreditScoreRequest; about to exit CustomMediation1 |  | Activ |

- 5. Review the fine-grained trace of the mediation flow component.
- a. Examine the results of the fine-grained trace in the **Events** window.

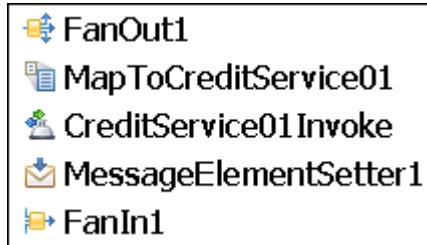


Your Event Trace might not show in the same order as shown here.

The trace shows the mediation primitives that are run during the mediation, in the sequence in which they were run. (Not all of the trace is shown here.)

The three primitives that are associated with the first credit score service (transform, invoke, message setter) follow the `FanOut1` primitive. `FanIn1` follows the three credit

score service primitives.



You specified that the fan-in must receive three messages before it would fire its **out** terminal. Because the fan-in decision point was not yet reached, the aggregation begins again. It sends a copy of the original input message to the set of primitives that are associated with the second credit score service. It then sends another copy for the third credit score service. At that point, the fan-in decision point is reached, and the fan-in sends the input message to the custom mediation primitive and through the rest of the mediation.

- \_\_ b. Click **CreditService01Invoke** in the event trace, and examine the **body** of the mediation message contents (on the right side of the pane).

|                              |                         |          |
|------------------------------|-------------------------|----------|
| body                         | calculateCreditScore... | [ab]     |
| calculateCreditScoreResponse | calculateCreditScore... | [ab]     |
| calculateCreditScoreReturn   | CreditCheckRequest      | [ab]     |
| dateRequested                | String                  | [ab]     |
| accountNumber                | String                  | [ab]     |
| creditScore                  | int                     | [ab] 11  |
| companyName                  | String                  | [ab] IBM |

The credit score is returned from the first service.

Now click **MessageElementSetter1** in the event trace, and examine the mediation message. You can see that the result that is returned from **CreditService01Invoke** is stored in the shared context.

|                       |                       |          |
|-----------------------|-----------------------|----------|
| context               | ContextType           | [ab]     |
| shared                | CreditCheckResultS... | [ab]     |
| CreditService03Result | CreditCheck           | [ab]     |
| CreditService02Result | CreditCheck           | [ab]     |
| CreditService01Result | CreditCheckRequest    | [ab]     |
| dateRequested         | String                | [ab]     |
| accountNumber         | String                | [ab]     |
| creditScore           | int                   | [ab] 11  |
| companyName           | String                | [ab] IBM |

- \_\_\_ c. Similarly, review the credit scores that the **CreditService02Invoke** and **CreditService03Invoke** returned.

- \_\_\_ d. Click **FanIn1** just before **CustomMediation1** in the event trace, and examine the shared context in the mediation message. **FanIn1** occurs just before **CustomMediation1**. You can see that all three results that are returned from the three credit score service invocations were stored.

| context               |                       | ContextType           |
|-----------------------|-----------------------|-----------------------|
|                       | shared                | CreditCheckResultS... |
|                       | CreditService03Result | CreditCheckRequest3   |
| companyName           | String                | IBM                   |
| creditScore           | int                   | 9                     |
| dateRequested         | String                |                       |
| accountNumber         | String                |                       |
| CreditService02Result | CreditCheckRequest2   |                       |
| companyName           | String                | IBM                   |
| dateRequested         | String                |                       |
| accountNumber         | String                |                       |
| creditScore           | int                   | 10                    |
| CreditService01Result | CreditCheckRequest    |                       |
| creditScore           | int                   | 11                    |
| dateRequested         | String                |                       |
| companyName           | String                | IBM                   |
| accountNumber         | String                |                       |

Your shared context might not show in the same order as shown here.

- \_\_\_ e. Click **CustomMediation1** in the trace, and then review the credit score that is returned in the **body** of the mediation message. The logic of the custom mediation determines the lowest of the credit score values that are returned from the three service invocations, and then returns that score in the business object.

|                              |                        |
|------------------------------|------------------------|
| body                         | calculateCreditScor... |
| calculateCreditScoreResponse | calculateCreditScor... |
| calculateCreditScoreReturn   | CreditCheckRequest     |
| dateRequested                | String                 |
| accountNumber                | String                 |
| creditScore                  | int                    |
| companyName                  | String                 |

- \_\_\_ 6. If you want, review the custom mediation logic to determine how the determination and assignments are made.
- \_\_\_ a. In the **Projects** view, expand the **CreditScoreMediationService**.
- \_\_\_ b. Double-click **Assembly Diagram**. The assembly editor opens.

- c. Double-click the **CreditScoreMediation** component.
  - d. Click the **getLowest** operation. The mediation flow editor opens.
  - e. Right-click the **CustomMediation1** primitive, and select **Show In > Properties View** from the menu.
  - f. Click the **Details** tab and review the Java code, but do not modify it.
- 



## Questions

This exercise assumes that each of the three credit score service invocations always returns a response. What would happen if one or more do not, or if the invocation has a long delay?

Hint: The fan-in primitive contains a timeout property that causes the flow to send the SMO to the **timeout** terminal instead of the **out** terminal. You might also find it necessary to use the **fail** terminal of the service invoke primitives.

---

## Part 5: *Cleaning up the workspace*

- 1. Remove the deployed projects from the server.
  - a. In the **Servers** view, right-click **IBM Process Server v8.5.7 at localhost**.
  - b. Select **Add and Remove** from the menu.
  - c. Click **Remove All**.
  - d. Click **Finish**. The projects are removed to the server. Wait for deployment to complete. The server shows **[Started, Synchronized]** when deployment is complete. It might take several minutes.
- 2. Close the IBM Integration Designer. Click **File > Exit**, or click **X** in the upper-right corner.

## End of exercise

---

# Exercise 11. Writing a generic error handler for IBM Process Server

## Estimated time

02:00

## Overview

In this exercise, you write generic error handler logic for IBM Process Server in a mediation subflow. The subflow handles runtime failures and logs the messages to the console.

## Objectives

After completing this exercise, you should be able to:

- Write a mediation subflow
- Implement mediation logic to manage runtime errors
- Use mediation primitives to control the flow of execution in a mediation, including the message filter and flow order primitives
- Implement the custom mediation primitive

## Introduction

Frequently, you want to have a standard behavior to handle runtime exceptions regardless of failure point. In this exercise, you are going to learn how to define a mediation subflow that defines a standard behavior (or pattern) to handle errors. A mediation subflow is a preconfigured set of mediation primitives that are wired together to realize a common pattern or use case. Mediation subflows are run in the context of a parent flow, and can be reused in mediation flows or in subflows.

## Requirements

Completing the exercise requires a lab environment that includes the exercise support files, Mozilla Firefox, IBM Business Process Manager V8.5.7 Advanced, IBM DB2 V10.1, and the IBM Process Server V8.5.7 test environment.

## Exercise instructions

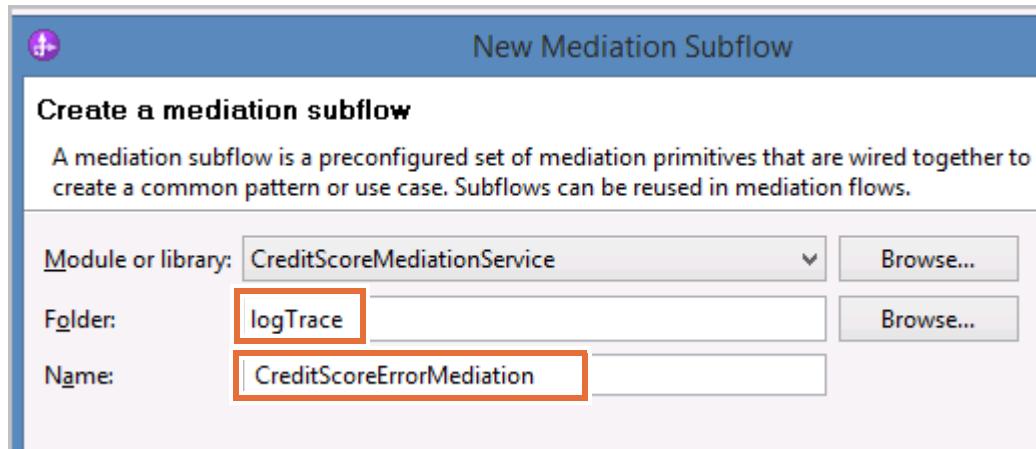
A credit mediation service is being used to determine the best credit score for a customer. The service returns the best credit score result to the account verification process. However, the mediation service does not contain any error handling. During the mediation process, if the best credit score service fails, it throws an exception and interrupts the process. In this exercise, you are going to implement general-purpose error handling logic as a mediation subflow.

A mediation subflow is a preconfigured set of mediation primitives that are wired together to realize a common pattern or use case. Mediation subflows are run in the context of a parent flow, and can be reused in mediation flows or in subflows. Therefore, implementing error handling logic in a subflow provides reusable and consistent error handling in an application. Later in this exercise, you wire it into the mediation.

### **Part 1: Implement the generic error handler subflow**

In this section of the exercise, you are going to create a mediation module to contain the credit service error handling subflow.

- \_\_\_ 1. Open the Exercise 11 workspace.
  - \_\_\_ a. On your desktop, open the folder that is labeled **Exercise Shortcuts**.
  - \_\_\_ b. Double-click the shortcut that is labeled **Exercise 11**.
  - \_\_\_ c. Close the **Getting Started** tab.
- \_\_\_ 2. Create a mediation subflow called `CreditScoreErrorMediation`.
  - \_\_\_ a. In the Business Integration view, expand `CreditScoreMediationService`.
  - \_\_\_ b. Right-click the **Integration Logic**, and click **New > Mediation Subflow**.
  - \_\_\_ c. Accept the default module name.
  - \_\_\_ d. Set **Folder** to: `logTrace`
  - \_\_\_ e. Enter `CreditScoreErrorMediation` in the **Name** field.



- \_\_\_ f. Click **Finish**.

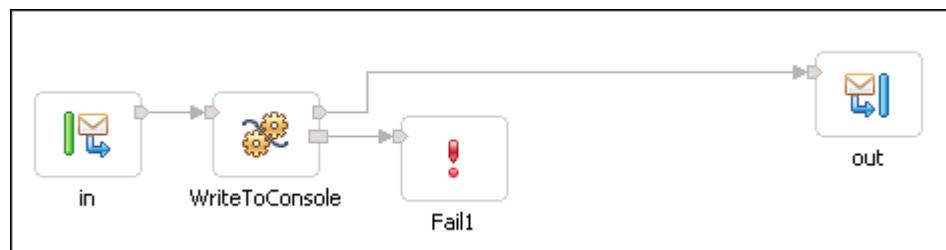
- \_\_\_ 3. The CreditScoreErrorMediation subflow editor should be open. If not, double-click **CreditScoreErrorMediation** in the **Mediation Subflows > logTrace** folder in the Business integration view. The **in** and **out** nodes are shown in the subflow editor.
- \_\_\_ 4. Add a custom mediation primitive named `WriteToConsole` to the subflow implementation.
  - \_\_\_ a. Select **Custom Mediation** from the **Transformation** drawer in the palette, and click anywhere in the editor. The primitive is shown in the canvas with a default name.



- \_\_\_ b. Change the name of the custom mediation to: `WriteToConsole`
- \_\_\_ c. Wire the **out** terminal of the **in** node to the **in** terminal of the `WriteToConsole` primitive, and wire the **out** terminal of the `WriteToConsole` primitive to the **in** terminal of the **out** node.



- \_\_\_ d. In the **Error Handling** drawer in the palette, select the **Fail** primitive and then click anywhere in the editor. The fail primitive is shown.
- \_\_\_ e. Accept the default name for the fail primitive.
- \_\_\_ f. Wire the **fail** terminal of the `WriteToConsole` primitive to the `Fail1` primitive.



- \_\_\_ 5. Implement the `WriteToConsole` logic to print the failure message to the console.
  - \_\_\_ a. Right-click `WriteToConsole` and select **Show In > Properties View**.
  - \_\_\_ b. In the Properties view, select the **Details** tab. By default, some skeleton Java code is in the custom mediation primitive. This code fires the service message object to the output terminal.
  - \_\_\_ c. Open Windows Explorer to: `c:\labfiles\Support_Files\EX11`
  - \_\_\_ d. Open `WriteToConsole.txt` in a text editor such as WordPad.
  - \_\_\_ e. Copy the contents of `WriteToConsole.txt` and paste it into the custom mediation implementation. Be sure to copy over all of the contents in the implementation code.

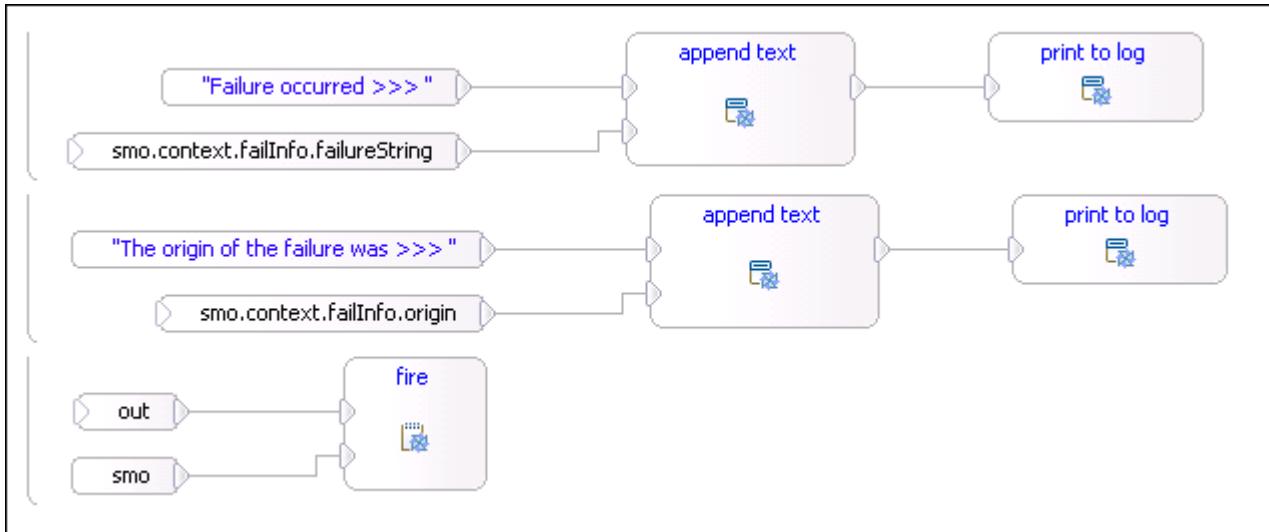
— 6. Save your changes.



### Optional

Instead of copying and pasting the Java code into the custom mediation implementation, you can optionally build the code with the Java visual editor. If you already copied and pasted the Java code, then you **do not** need to do the following visual steps. The steps are listed for your reference and you can review them. When you are done reviewing, then go to **Part 2: Implement the generic subflow into a mediation flow** of this exercise, and continue with the steps.

When complete, the visual Java code looks like the following diagram:

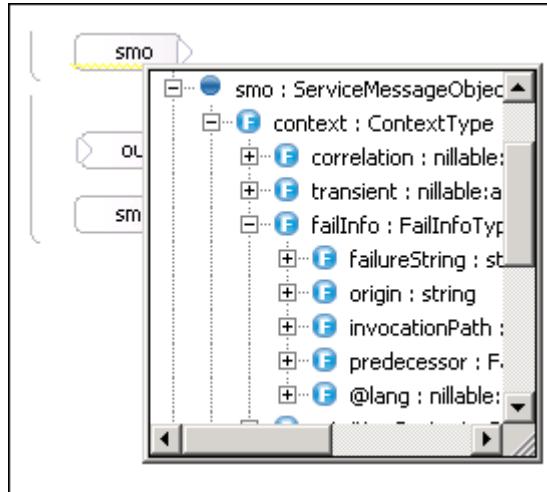


1. Select the **Visual** implementation. When the Question window opens, click **Yes** to switch to the visual editor.
2. Select **smo** from the **Inputs** list on the right, and drag it above the existing automatically generated snippets.



3. Click inside the **smo** snippet element, and expand **smo:ServiceMessageObject > context:ContextType > failInfo:FailInfoType**.

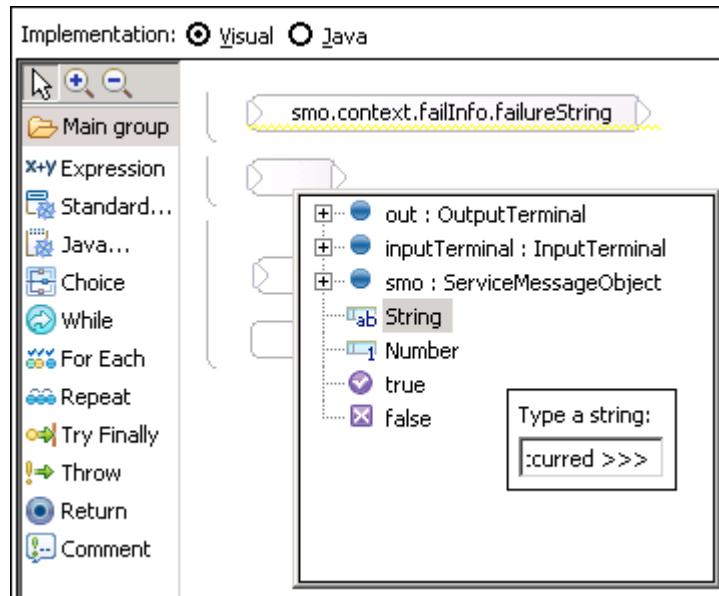
4. Select **failureString:string**.



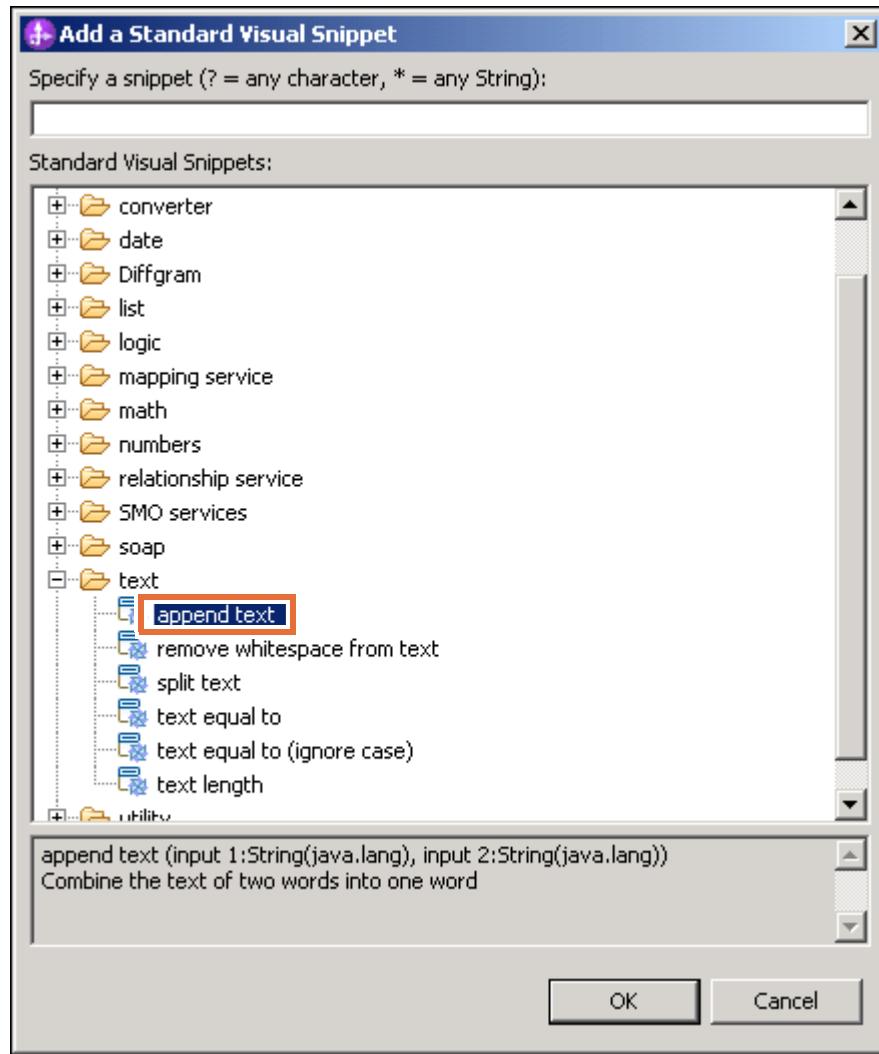
5. Select **x+y Expression** from the palette on the left, and drop it on the `WriteToConsole` visual area.  
 6. Click inside the **x+y Expression**, and select **String**. Enter the text:

`Failure occurred >>>`

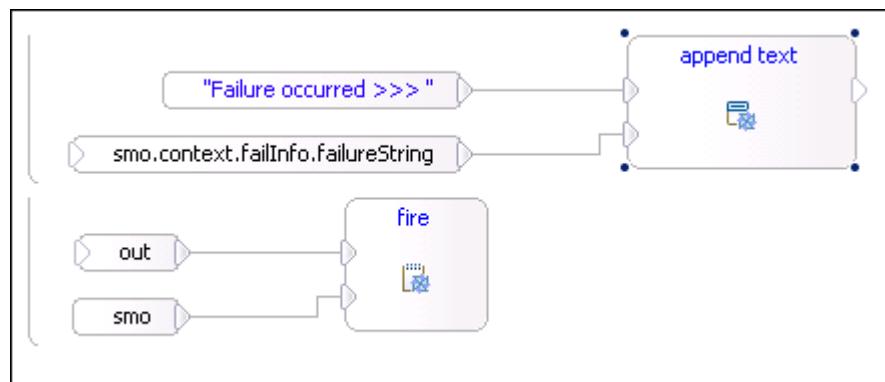
inside the **Type a string** box. Press Enter when you are done. You do not have to add quotation marks; the visual editor adds them automatically.



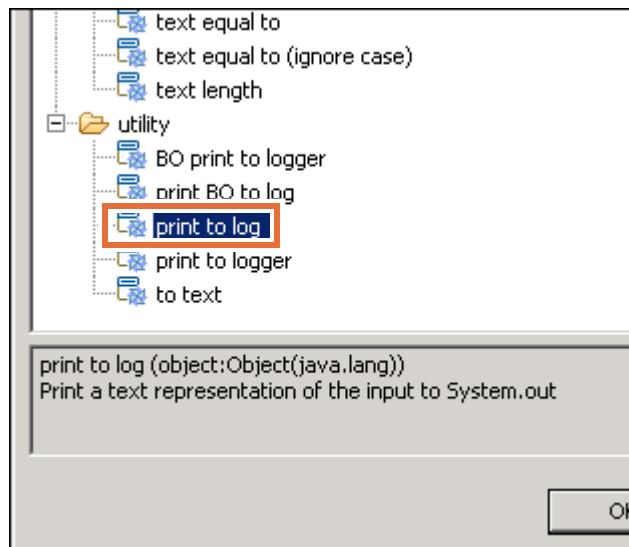
7. Click **Standard Visual Snippet** from the palette on the left. The **Add a Standard Visual Snippet** window opens. Expand **text**, and select **append text**.



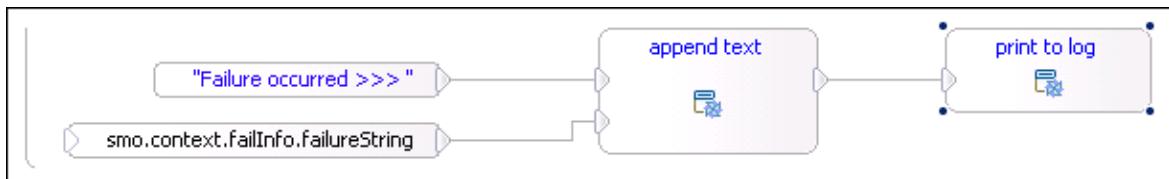
8. Click **OK**, and click above the "Failure occurred >>>" expression.  
 9. Wire the "Failure occurred >>>" expression and `smo.context.failInfo.failureString` to the **append text** snippet.



10. Again, click **Standard Visual Snippet** from the palette on the left. In the **Add a Standard Visual Snippet** window, expand **utility**, and select **print to log**. Click **OK**.



11. Click under the **append text** icon to add the **print to log** snippet to the editor.  
12. Wire the **append text** snippet to the **print to log** snippet.



13. Now, repeat the steps to add more logic to print the failure origin to the console. Select **smo** from the variable list on the right, and drag it above the existing automatically generated snippets.  
14. Click inside the **smo** snippet element, and expand **smo:ServiceMessageObject > context:ContextType > failInfo:FailInfoType**. Select **origin:string**.  
15. Select **x+y Expression** from the palette on the left, and drop it on the **WriteToConsole** visual area, below the **smo.context.failInfo.origin** icon that you created in the previous step.  
16. Click inside the **x+y Expression**, and select **String**. Enter the text:

The origin of the failure was >>>

inside the **Type a string** box. Press Enter. The quotation marks are added automatically.



### Hint

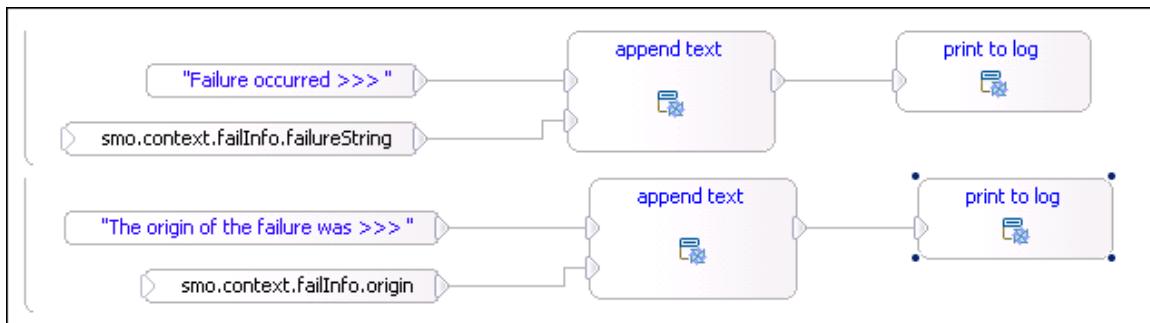
Alternatively, you can type in "The origin of the failure was >>>" inside the **x+y Expression**. Both methods generate the same code.

17. Click **Standard Visual Snippet** from the palette on the left, expand **text**, and then select **append text**.

18. Click **OK** and then click above the "The origin of the failure was >>>" expression.
19. Wire the "The origin of the failure was >>>" expression and **smo.context.failInfo.origin** to the append text snippet.



20. Again, click **Standard Visual Snippet** from the palette on the left. Expand **utility**, and select **print to log**.
21. Click **OK** and click under the **append text** icon that you just created to add the **print to log** snippet to editor. Wire the **append text** snippet to the **print to log** snippet. Your visual snippet editor resembles the following screen capture.



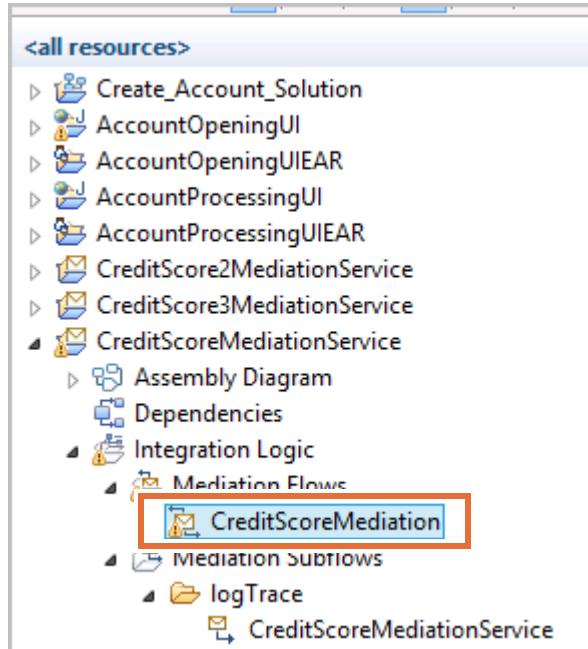
22. Save your changes.

## **Part 2: Implement the generic subflow into a mediation flow**

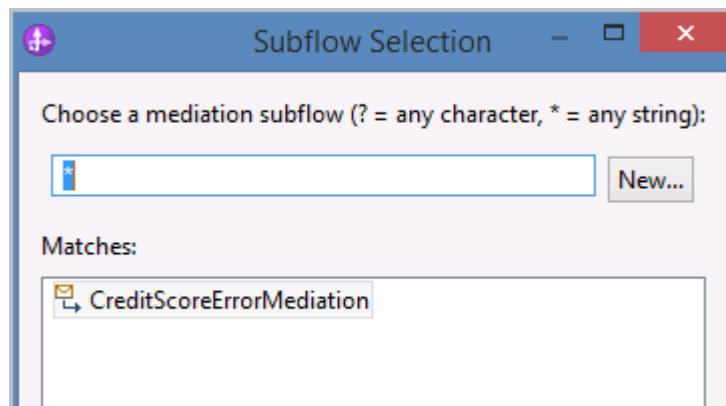
The generic error handling subflow is used in the credit score service mediation flow. In this section of the exercise, you wire the subflow into the mediation flow.

- 1. In the Business Integration view, expand **CreditScoreMediationService > Integration Logic > Mediation Flows**.

- \_\_ 2. Double-click **CreditScoreMediation** to open it in the mediation editor.



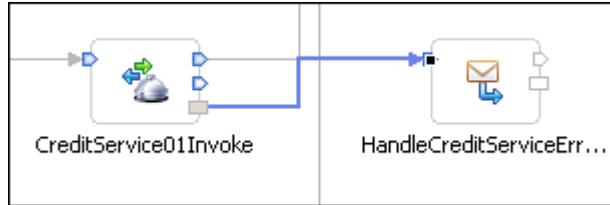
- \_\_ 3. If you see a tip window, check the **Do not show tips** check box, and close the window.
- \_\_ 4. In the **Operation Connections** pane, click the `getLowest` operation. The mediation flow editor opens.
- \_\_ 5. Add the error mediation subflow to the diagram.
  - \_\_ a. Select the **Subflow** primitive from the **Mediation Subflow** drawer and add it to the canvas. The **Subflow Selection** screen is shown.
  - \_\_ b. Select **CreditScoreErrorMediation** and click **OK**.



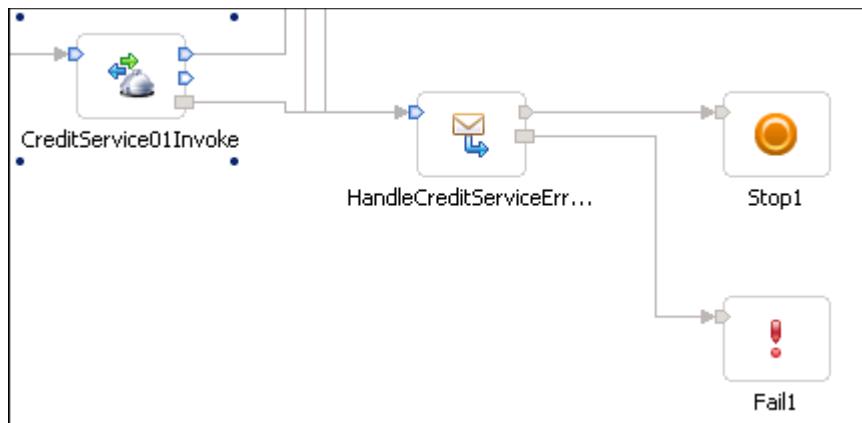
- \_\_ c. Change the name of the subflow to: `HandleCreditServiceErrors`



- \_\_\_ d. Press Ctrl+S to save the changes. If you are prompted to save the workspace in simple XML format, check the **Remember my decision** check box and click **No**.
- \_\_\_ e. Wire the **fail** terminal of the `CreditService01Invoke` primitive to the **in** terminal of the subflow primitive.

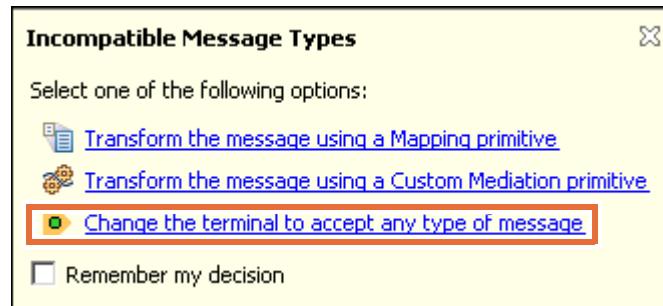


- \_\_\_ f. From the palette, in the **Error Handling** drawer, select a **Stop** primitive and add it to the canvas. Accept the default name.
- \_\_\_ g. Similarly, select a **Fail** primitive from the palette and add it to the canvas. Accept the default name.
- \_\_\_ h. Wire the **out** terminal of the `HandleCreditServiceErrors` to the **in** terminal of the **Stop1** primitive.
- \_\_\_ i. Wire the **fail** terminal of the `HandleCreditServiceErrors` to the **in** terminal of the **Fail1** primitive.

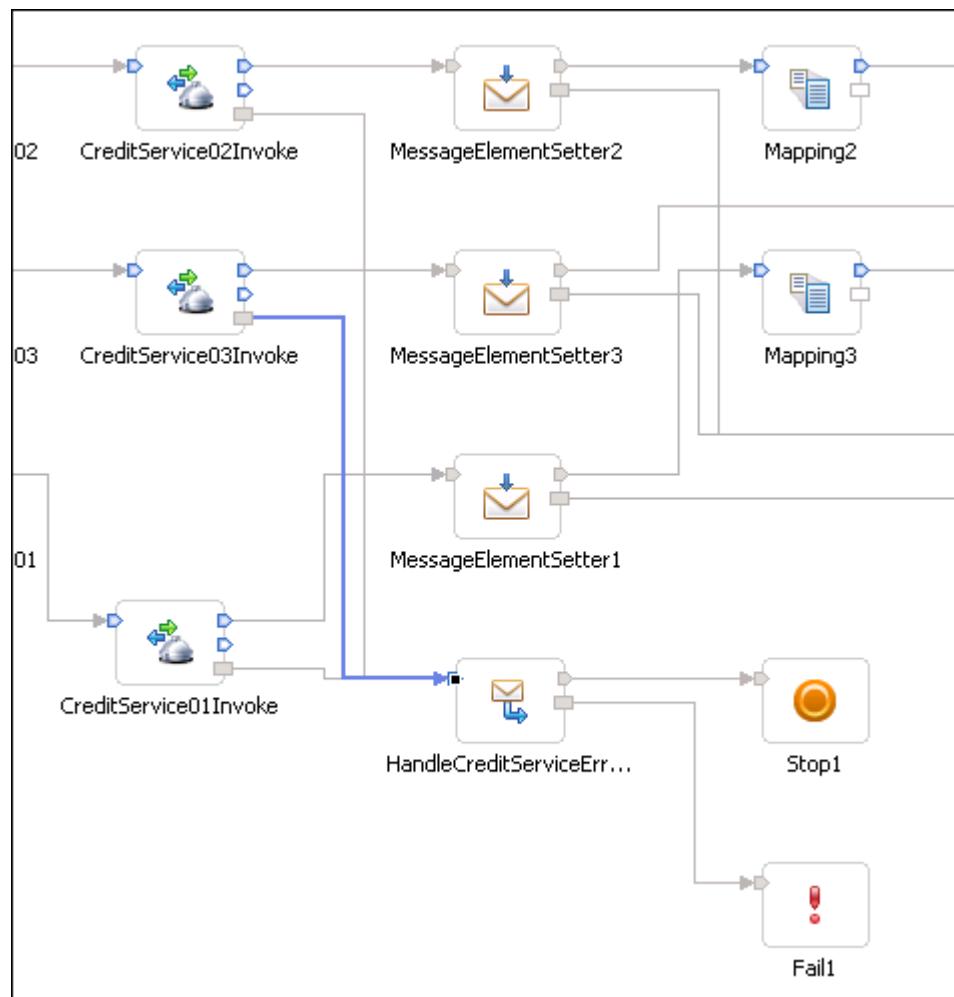


- \_\_\_ j. Press Ctrl+S to save the changes.
- \_\_\_ 6. Similarly, connect the remaining service invocation primitives to the error handler.
  - \_\_\_ a. Wire the **fail** terminal of the `CreditService02Invoke` primitive to the **in** terminal of the subflow primitive.
  - \_\_\_ b. An **Incompatible Message Types** window opens. When you wired the first service invocation to the **in** terminal, the type was automatically set. When you wire the consecutive primitives into the same terminal, the types are different.

However, in this case it is not necessary to use a transformation. Click **Change the terminal to accept any type of message**.



- c. Wire the **fail** terminal of the `CreditService03Invoke` primitive to the **in** terminal of the subflow primitive.

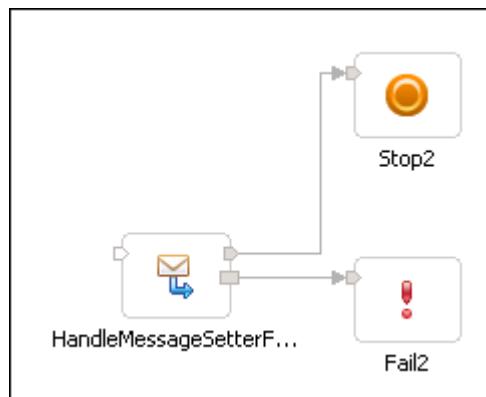


- 7. Save your changes.
- 8. Add a second subflow primitive to capture failure messages from the message element setter primitives.
  - a. From the **Mediation Subflow** drawer, select the **Subflow** primitive and add it to the canvas. The **Subflow Selection** window is shown.
  - b. Select the `CreditScoreErrorMediation` subflow and click **OK**.

- \_\_\_ c. Set the name of the subflow primitive to: HandleMessageSetterFailure

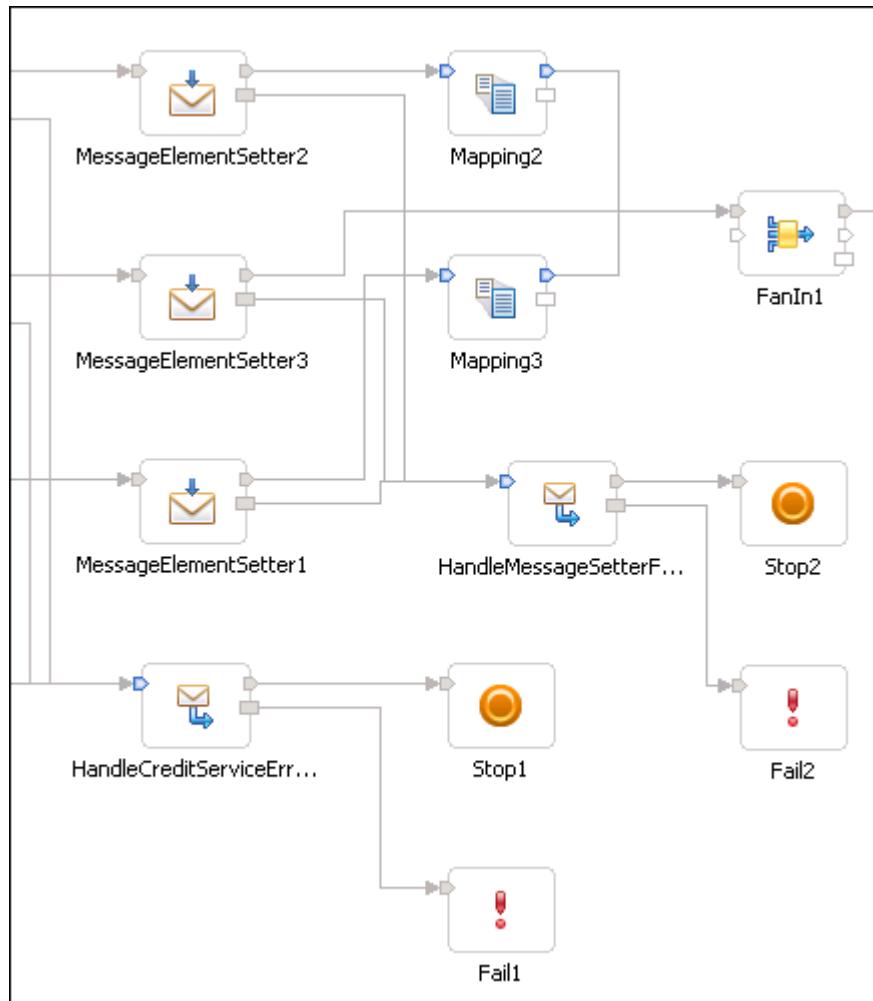


- \_\_\_ 9. From the **Error Handling** drawer, add a **Stop** and a **Fail** primitive to the diagram. Accept the default names.
- \_\_\_ 10. Wire the **out** terminal of the `HandleMessageSetterFailure` primitive to the **in** terminal of the **Stop2** primitive.
- \_\_\_ 11. Wire the **fail** terminal of the `HandleMessageSetterFailure` primitive to the **in** terminal of the **Fail2** primitive.



- \_\_\_ 12. Wire the message element setter primitives in to the new generic error handler.
  - \_\_\_ a. Wire the **fail** terminal of the `MessageElementSetter1` primitive to the **in** terminal of `HandleMessageSetterFailure`.
  - \_\_\_ b. Wire the **fail** terminal of the `MessageElementSetter2` primitive to the **in** terminal of `HandleMessageSetterFailure`.
  - \_\_\_ c. In the **Incompatible Message Types** window, click **Change the terminal to accept any type of message**.

- \_\_\_ d. Wire the **fail** terminal of the `MessageElementSetter3` primitive to the **in** terminal of `HandleMessageSetterFailure`.



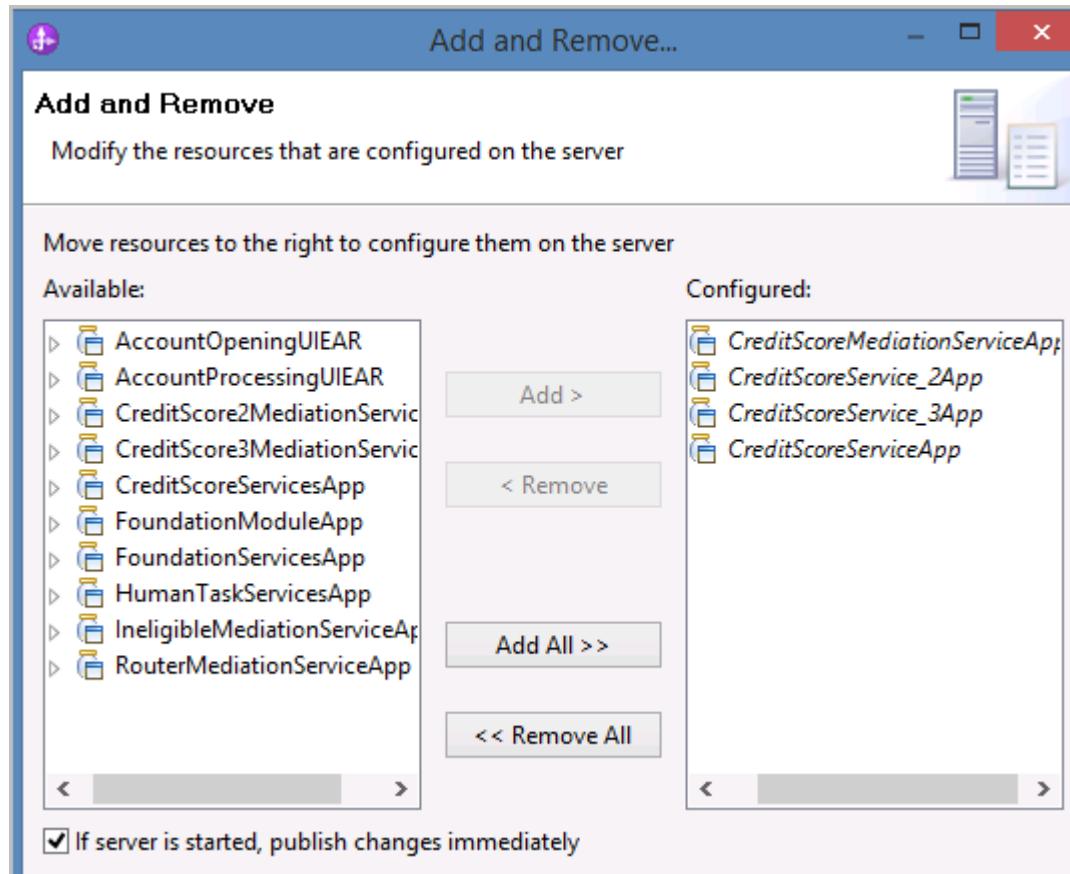
- \_\_\_ 13. Right-click the flow diagram and select **Automatic Layout**. This option formats the diagram for legibility.  
 \_\_\_ 14. Press Shift+Ctrl+S to save all changes.

### **Part 3: Test the mediation flow with ErrorHandlingSubflow**

In this section of the exercise, you test the credit score mediation service flow. This flow tests the error handling subflow.

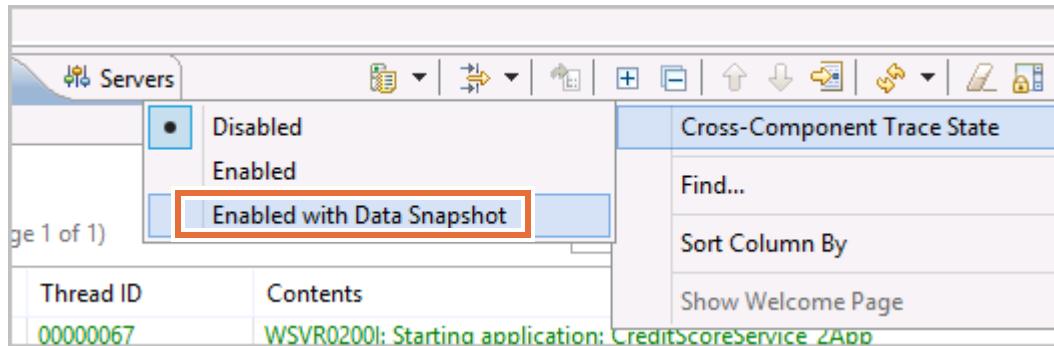
- \_\_\_ 1. Start the server unless it is already running, and deploy the mediation flow to the server.
  - \_\_\_ a. In the **Servers** tab, select **IBM Process Server v8.5.7 at localhost** and click the **Start** icon (the green arrow). Alternatively, you can right-click the server, and select **Start** from the menu.
  - \_\_\_ b. After the server state changes to **Started, Synchronized**, right-click **IBM Process Server v8.5.7 at localhost** and select **Add and Remove**.

- \_\_\_ c. From the **Available** list, select `CreditScoreMediationServiceApp`, `CreditScoreService_2App`, `CreditScoreService_3App`, `CreditScoreServiceApp`, and click **Add**.



- \_\_\_ d. Click **Finish**. While the project is being deployed, the server state shows Started, Publishing.
  - \_\_\_ e. Wait until the application is fully deployed. The **Servers** view shows that the server state shows Started, Synchronized when the deployment is complete.
- \_\_\_ 2. Test `CreditScoreMediationExport`.
- \_\_\_ a. In the `CreditScoreMediationService` assembly diagram, right-click `CreditScoreMediationExport` and select **Test Component**.
  - \_\_\_ b. On the lower pane of the workbench, switch to the **Server Logs** view.

- \_\_\_ c. Enable cross-component trace by clicking the **View** menu icon (the rightmost icon on the server logs toolbar). Select **Cross-Component Trace State > Enable with Data Snapshot**.



- \_\_\_ d. In the Initial request parameters table, enter these initial test values:
- **accountNumber**: IBM007
  - **companyName**: IBM
  - **creditScore**: 0
  - **dateRequested**: 2014-09-12

| Initial request parameters:                                                    |             |                 |
|--------------------------------------------------------------------------------|-------------|-----------------|
| <input checked="" type="radio"/> Value editor <input type="radio"/> XML editor |             |                 |
| Name                                                                           | Type        | Value           |
| Input                                                                          | CreditCheck | [ab]            |
| accountNumber                                                                  | string      | [ab] IBM007     |
| companyName                                                                    | string      | [ab] IBM        |
| creditScore                                                                    | int         | [ab] 0          |
| dateRequested                                                                  | string      | [ab] 2014-09-12 |



### Note

The request parameter values can be any valid company name, such as IBM, ACME, AbcCo, or TestCo.

- \_\_\_ e. In the Events pane, click **Continue** to initiate the test.
- \_\_\_ f. If you are prompted to enter the deployment location, select **IBM Process Server v8.5.7 at localhost**, and click **Finish**.
- \_\_\_ g. If you are prompted to enter user credentials, use **User Name** admin and **Password** web1sphere and then click **OK**.

- \_\_ h. The test completes successfully with a credit score of 9.

| Return parameters:      |                              |                              |
|-------------------------|------------------------------|------------------------------|
| Value Editor XML Source |                              |                              |
|                         | Name                         | Type                         |
|                         | Output                       | calculateCreditScore... [ab] |
|                         | calculateCreditScoreReturn * | CreditCheckRequest3 [ab]     |
|                         | accountNumber                | string [ab] IBM007           |
|                         | companyName                  | string [ab] IBM              |
|                         | creditScore                  | int [ab] 9                   |
|                         | dateRequested                | string [ab] 2014-09-12       |

- \_\_ 3. Examine the events of the test.  
\_\_ a. Maximize the Integration Test Client view.

- \_\_\_ b. The request is sent from the export SCA component to the mediation flow component. The mediation flow component starts the mediation service, which calls the three credit service web services, collects the results, and aggregates the responses with the **FanIn** primitive.

**Integration Test Client: CreditScoreMediationService\_Test**

**Events**

This area displays the events in a test trace. Select an event to display its properties in the General Properties and Detailed Properties sections. [More...](#)

The screenshot shows the 'Events' tab of the Integration Test Client. A tree view displays a sequence of events: an initial 'Invoke' (CreditScoreMediationExport:getLowest), followed by an 'Invoke started' event, then a 'Binding' event (SCA:CreditScoreMediationExport). This is followed by another 'Invoke' event (CreditScoreMediationExport:getLowest), a 'Request' event (CreditScoreMediationExport --> CreditScoreMediation:getLowest), and finally a 'Fine-Grained Trace' event (CreditScoreMediation:CreditScoreMediation). The trace details the internal processing steps, including 'getLowest : CreditScoreAggregate', 'FanOut1', 'MapToCreditService01', 'CreditService01Invoke', 'MessageElementSetter1', 'Mapping3', 'FanIn1', 'MapToCreditService02', 'CreditService02Invoke', 'MessageElementSetter2', 'Mapping2', 'FanIn1', 'MapToCreditService03', and 'CreditService03Invoke'. At the bottom, there are tabs for 'Events' (which is selected) and 'Configurations'.

- \_\_\_ c. Restore the Integration Test Client view.
4. Examine the server logs.
- \_\_\_ a. Maximize the Server Logs view.

- \_\_\_ b. To begin the test, the export SCA component started the getLowest operation on the mediation flow component. The mediation flow started three web services to solicit and aggregate the responses. Each of the four invocation sequences is available for inspection in the Server Logs.

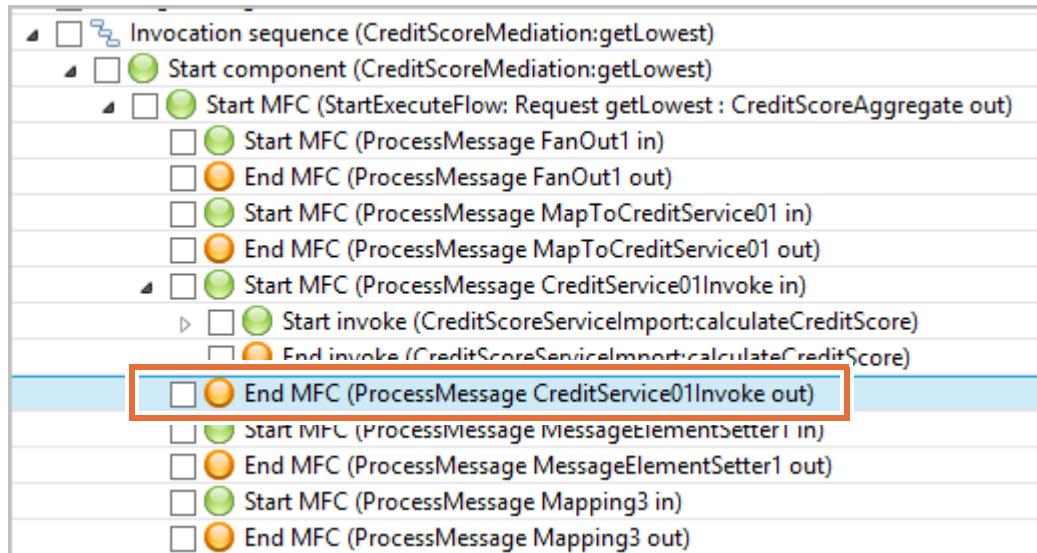
| Type                                                                                       |
|--------------------------------------------------------------------------------------------|
| <input type="checkbox"/> Log message                                                       |
| ▶ <input type="checkbox"/> Invocation sequence (CreditScoreMediation:getLowest)            |
| ▶ <input type="checkbox"/> Invocation sequence (CreditScoreRG:calculateCreditScore)        |
| ▶ <input type="checkbox"/> Invocation sequence (CreditScoreRG:calculateCreditScore)        |
| ▶ <input type="checkbox"/> Invocation sequence (FilterOutBadRequests:calculateCreditScore) |

- \_\_ c. Expand the first invocation sequence (CreditScoreMediation:getLowest). This sequence started the mediation flow component (“MFC”). Feel free to adjust the column width to view the full details.
  - d. Expand **Start component > Start MFC**.

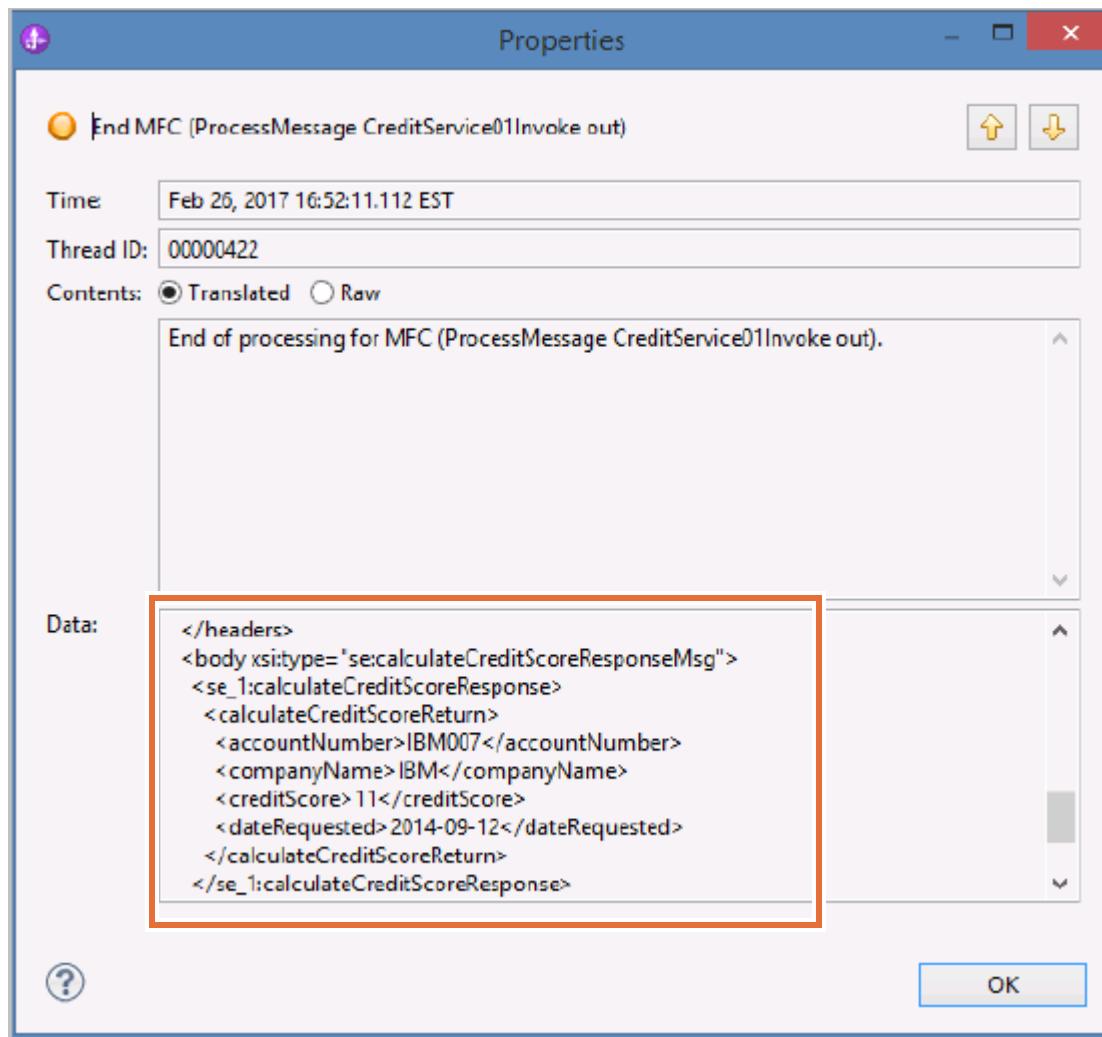
| Type                                                                                                           |
|----------------------------------------------------------------------------------------------------------------|
| <input type="checkbox"/> Log message                                                                           |
| <input type="checkbox"/> Log message                                                                           |
| <input checked="" type="checkbox"/> Invocation sequence (CreditScoreMediation:getLowest)                       |
| <input checked="" type="checkbox"/> Start component (CreditScoreMediation:getLowest)                           |
| <input checked="" type="checkbox"/> Start MFC (StartExecuteFlow: Request getLowest : CreditScoreAggregate out) |
| <input checked="" type="checkbox"/> Start MFC (ProcessMessage FanOut1 in)                                      |
| <input checked="" type="checkbox"/> End MFC (ProcessMessage FanOut1 out)                                       |
| <input checked="" type="checkbox"/> Start MFC (ProcessMessage MapToCreditService01 in)                         |
| <input checked="" type="checkbox"/> End MFC (ProcessMessage MapToCreditService01 out)                          |
| <input checked="" type="checkbox"/> Start MFC (ProcessMessage CreditService01Invoke in)                        |
| <input checked="" type="checkbox"/> End MFC (ProcessMessage CreditService01Invoke out)                         |
| <input checked="" type="checkbox"/> Start MFC (ProcessMessage MessageElementSetter1 in)                        |
| <input checked="" type="checkbox"/> End MFC (ProcessMessage MessageElementSetter1 out)                         |
| <input checked="" type="checkbox"/> Start MFC (ProcessMessage Mapping3 in)                                     |

- \_\_ e. Examine the log entries of the mediation flow component and compare them to the events that you examined in the integration test client. With the cross-component trace, you can examine each primitive in the mediation flow.

- f. Double-click the `End MFC (ProcessMessage CreditService01Invoke out)` log entry. The data snapshot parameter of cross-component trace captured the state of the data during execution.



- \_\_ g. In the Data section, scroll down to the body of the message. Examine the value of creditScore in calculateCreditScoreReturn. Click **OK** to close the Properties window.



- \_\_ h. Double-click the End MFC (ProcessMessage CreditService02Invoke out) and End MFC (ProcessMessage CreditService03Invoke out) log entries and compare the values.



## Questions

The lowest return value for **creditScore** with company name IBM was 9. Which service returned that value?

**Optional**

This test demonstrates the result of a successful flow. You can start the test again with other valid company name values, such as ACME, AbcCo, and TestCo.

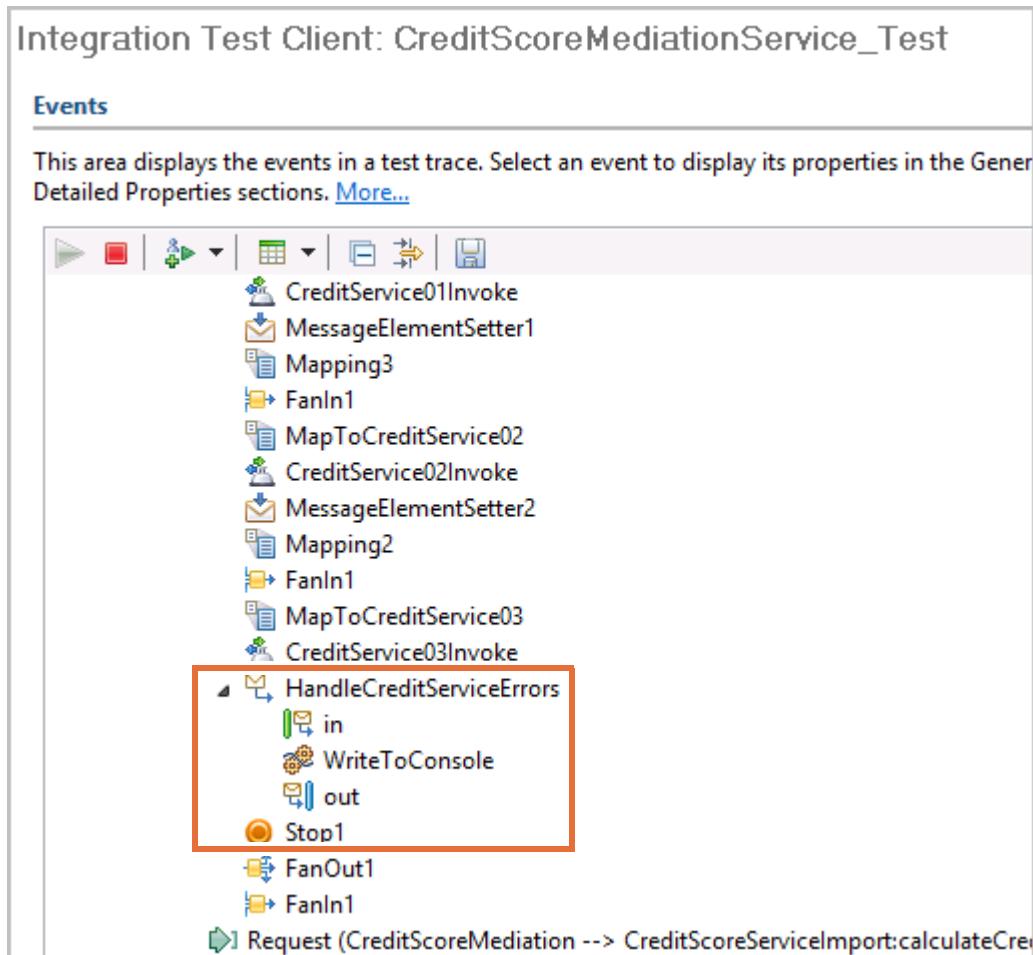
- 5. Test the flow with an invalid company name. The submap that you built is to handle generic errors. In this test, any company name that is not IBM, AbcCo, ACME, or TestCo is considered invalid.
  - a. Click **Clear** in the **Server Logs** view. This step is not necessary, but with this step you can focus on the results of the new test.
  - b. In the integration test client, begin another test. You can close the current **CreditScoreMediationService\_Test** and start a new test.
  - c. Change the values in the **Initial request parameters** table to:
    - **accountNumber**: HAL999
    - **companyName**: HAL
    - **creditScore**: 0
    - **dateRequested**: 2014-09-012

The screenshot shows the 'Value editor' interface with the 'Value editor' tab selected. It displays a table of initial request parameters:

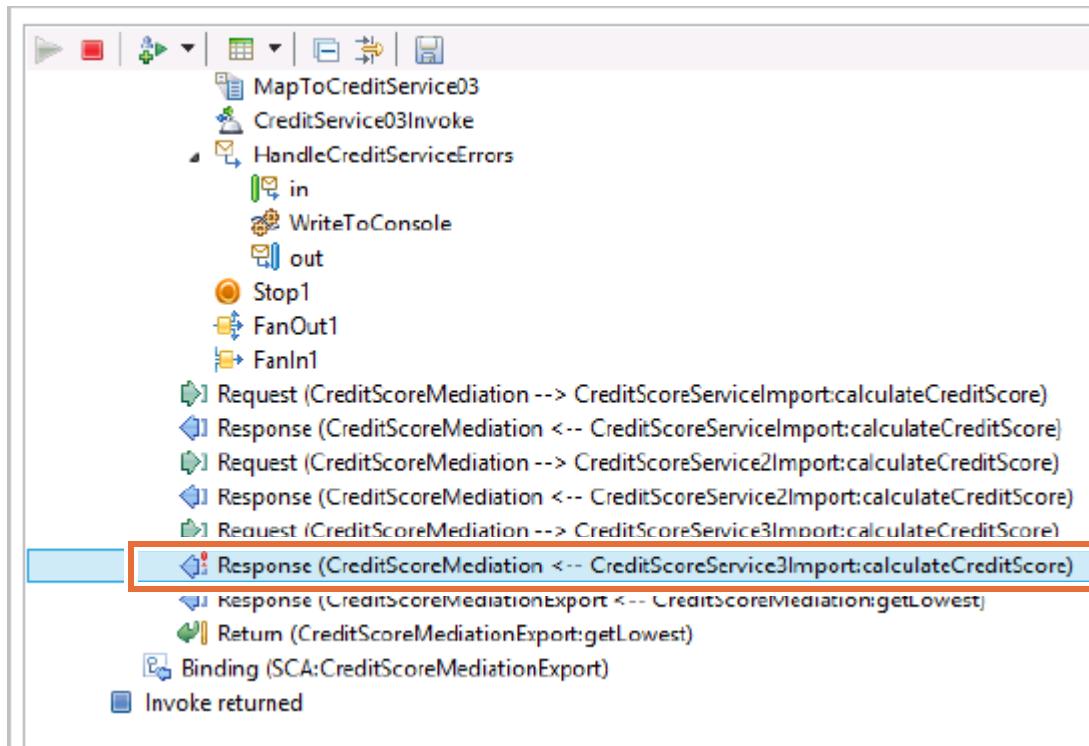
| Name          | Type        | Value         |
|---------------|-------------|---------------|
| Input         | CreditCheck | ab            |
| accountNumber | string      | ab HAL999     |
| companyName   | string      | ab HAL        |
| creditScore   | int         | ab 0          |
| dateRequested | string      | ab 2014-09-12 |

- d. In the Events pane, click **Continue** to initiate the test.
- e. The test completes with a value of **null**. Because the company name was invalid, an error is thrown. The generic error submap captured this fault.
- 6. Examine the captured events of the invalid test flow.
  - a. In the **Events** pane of the Integration Test Client, examine the web service invocations, which were used to determine the lowest credit score.

- \_\_\_ b. CreditService03Invoke triggered the generic error subflow. The subflow ran from start to finish and called a stop primitive.



- \_\_ c. Scroll down and click the invalid response (**CreditScoreMediation <-- CreditScoreService3Import:calculateCreditScore**).



- \_\_ d. Examine the fault message in the response table.

| Name         | Type    | Value                     |
|--------------|---------|---------------------------|
| faultMessage | Message |                           |
| message      | string  | Invalid company name: HAL |



## Questions

### Where did this fault message come from?

The `CreditService03Invoke` primitive has a web service binding to the `CreditScoreService_3` module. The `CreditScoreService_3` module contains an export component, which is wired to a business process named `FilterOutBadRequests`. This simple process accepts valid company names and throws a fault for invalid company names.

- \_\_ 7. Examine the contents of the **Server Logs** for the invalid company name.
- \_\_ a. Expand **Invocation sequence (FilterOutBadRequests:calculateCreditScore)**.

**Console (filtered): IBM Process Server v8.5.7 at localhost**

Show All Record Types (Hierarchical) > with only Server State and Error Contents (Page 1 of 1)

Type

- ▷   Invocation sequence (CreditScoreMediation:getLowest)
- ▷   Invocation sequence (CreditScoreRG:calculateCreditScore)
- ▷   Invocation sequence (CreditScoreRG:calculateCreditScore)
- ▷   **Invocation sequence (FilterOutBadRequests:calculateCreditScore)**
-  Log message

- \_\_ b. Expand **Start component (FilterOutBadRequests:calculateCreditScore) > Start BPEL process (FilterOutBadRequests)** and click **Log message**.

**Console (filtered): IBM Process Server v8.5.7 at localhost**

Show All Record Types (Hierarchical) > with only Server State and Error Contents (Page 1 of 1)

Type

- ▷   Invocation sequence (CreditScoreMediation:getLowest)
- ▷   Invocation sequence (CreditScoreRG:calculateCreditScore)
- ▷   Invocation sequence (CreditScoreRG:calculateCreditScore)
- ▷   **Invocation sequence (FilterOutBadRequests:calculateCreditScore)**
  - ▷   Start component (FilterOutBadRequests:calculateCreditScore)
    -  In BPEL process
    -  Start BPEL process (FilterOutBadRequests)
    -  End BPEL process (FilterOutBadRequests)
  - ▷   Start BPEL process (FilterOutBadRequests)
    -  Log message
    -  End BPEL process (FilterOutBadRequests)
    -  End component (FilterOutBadRequests:calculateCreditScore)
-  Log message

- \_\_ c. Because the company name was invalid, a fault is thrown.

|                                                                                      |
|--------------------------------------------------------------------------------------|
| End of processing for BPEL process FilterOutBadRequests: PI:90030159.97e330d4.a8...  |
| Start of processing for BPEL process FilterOutBadRequests: PI:90030159.97e330d4.a... |
| <b>CWWBE0061E: A fault 'InvalidCompanyName' was raised by activity 'Throw'.</b>      |
| End of processing for BPEL process FilterOutBadRequests: PI:90030159.97e330d4.a8...  |



## Information

In this exercise, the error handling logic is kept to minimal complexity. For a real-world implementation, you might choose to write the failure information to a database. Similarly, you can choose to accept between one and three valid responses in the **FanIn** primitive.

### **Part 4: Clean up the environment**

- 1. In the **Servers** view, right-click **IBM Process Server v8.5.7 at localhost** and click **Launch > Business Process Choreographer Explorer**.



## Note

You can also access the Business Process Choreographer Explorer from an IE browser by using the following URL:

<https://localhost:9443/bpc>

- 2. If you are prompted with security alerts, click **Yes**.
- 3. Log in to the Explorer with `admin` for the **User Name** and `web1sphere` for the **Password**.
- 4. Click **Process Instances > Administered By Me**.
- 5. Because the process instance with the invalid company name failed, it might be saved as an administration task. If no instance is listed, skip this step.

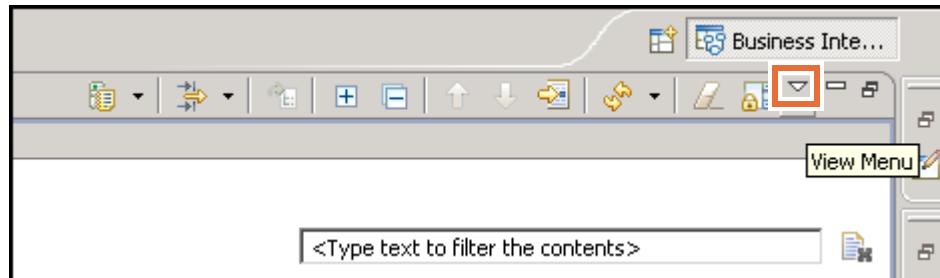
| Process Instances Administered By Me                                                                                                                                                                                                                                                                                               |                                                                 |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------|
| Use this page to work with process instances for which you are the process administrator. <a href="#">i</a>                                                                                                                                                                                                                        |                                                                 |
| <input type="button" value="Migrate"/> <input type="button" value="Terminate"/> <input type="button" value="Delete"/> <input type="button" value="Suspend"/> <input type="button" value="Resume"/> <input type="button" value="Restart"/> <input type="button" value="Compensate"/> <input type="button" value="Claim Ownership"/> |                                                                 |
| <input type="checkbox"/> Process Instance Name ◊<br><input checked="" type="checkbox"/> _PI:90030135.3f7180f7.70a8573f.edde04c1                                                                                                                                                                                                    | Process Template Name ◊<br><a href="#">FilterOutBadRequests</a> |
| Items found: 1 Items selected: 0                                                                                                                                                                                                                                                                                                   | Page 1 of 1 Items per page: <input type="button" value="2"/>    |

- 6. If you see an invalid instance, select the instance. Otherwise, if you see no instance, then go to step 10 to log out.
- 7. Click **Terminate**.
- 8. Select the invalid instance again.
- 9. Click **Delete**.

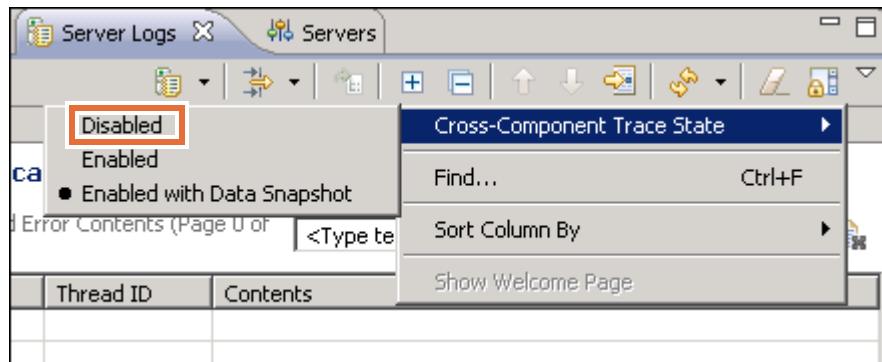
**Hint**

You cannot stop the `CreditScoreService_3` application until all process instances are stopped and removed.

- \_\_\_ 10. Click **Logout**.
- \_\_\_ 11. Remove the projects from the server.
  - \_\_\_ a. In the **Servers** view, right-click the server and click **Add and Remove**.
  - \_\_\_ b. Click **Remove All**.
  - \_\_\_ c. Click **Finish**. Wait for the server to remove the projects.
- \_\_\_ 12. Disable cross component trace.
  - \_\_\_ a. Access cross component trace settings by clicking **View Menu** at the far-right upper corner of the **Server Logs** view.



- \_\_\_ b. Disable cross component trace by selecting **Cross-Component Trace State > Disable** from the view menu.



- \_\_\_ 13. Close the IBM Integration Designer. Click **File > Exit**, or click **X** in the upper-right corner. You do not have to save the integration test client.

**End of exercise**

# Exercise 12. Implementing security

## Estimated time

01:30

## Overview

SOA provides flexibility and reusability of enterprise applications. In an earlier exercise, you explored an end-to-end scenario to send data from one EIS to another. The next phase of the development is to secure your application. Various web service proxy or gateway applications are available to tighten the security of your application. However, in this exercise, you learn how to set security properties on your application at development time.

## Objectives

After completing this exercise, you should be able to:

- Configure security qualifiers for an SCA application
- Configure role-based security for human tasks
- Configure authorization for business process components
- Use Business Process Choreographer Explorer to test human task and process permission settings
- Configure users and groups with Process Center internal security
- Configure participant groups
- Test human service security with Process Portal

## Introduction

In this exercise, you configure several security properties for IBM Process Server applications and Process Center applications. Security is a large and complex topic with many facets. The goal of this exercise is not to provide a comprehensive look at application and infrastructure security. Rather, the exercise introduces you to some of the security-related settings for IBM BPM applications.

## Requirements

To complete this exercise, you require a lab environment. This VMware image includes the exercise support files, Mozilla Firefox, IBM Integration Designer V8.5.7, IBM DB2 V10.1, RFHUTIL, WebSphere MQ V8.0, and the IBM Process Server V8.5.7 test environment.

## Exercise instructions

# Securing SCA applications

In this portion of the exercise, you are presented with an experience on how to turn on global security and set up the security QoS on your service interface.

### **Part 1: Preparing the workspace**

- 1. Open the Exercise 12 workspace.
  - a. On your desktop, open the folder that is labeled **Exercise Shortcuts**.
  - b. Double-click the shortcut that is labeled **Exercise 12** to start IBM Integration Designer.
  - c. Wait for the workspace build to complete. The workspace build might take a few minutes.
  - d. Close the **Getting Started** tab.
- 2. If IBM Process Server is not already running, start IBM Process Server.
  - a. In the **Servers** view, right-click **IBM Process Server v8.5.7 at localhost** and choose **Start** from the menu.
  - b. The server start is complete when the **Status** changes to **Started**, and you see the message **Server server1 open for e-business** in the **Server Logs** view.

### **Part 2: Creating user groups and users**

In this part of the exercise, you run a customized script that creates several users and groups that you use to test the components in this exercise.



#### Information

This script is available at the `C:\labfiles\Support Files\EX12` location. It is also available at: <http://publib.boulder.ibm.com/bpcsamp/humanTaskFeatures/singlePersonWorkflow.html>

The script creates the following users:

- approverA
- approverB
- approverC
- bpadminA
- bpadminB
- supervisorA
- supervisorB

The script also creates the following groups:

- bpadmins: Contains bpadminA and bpadminB
  - approvers: Contains approverA, approverB, and approverC
  - supervisors: Contains supervisorA and supervisorB
- 1. On the Windows desktop, right-click the Windows logo icon and click **Run** from the menu.
  - 2. In the **Run** window, enter: cmd
  - 3. Click **OK**.
  - 4. In the command window, enter the following command to change directories:  
cd C:\labfiles\Support Files\EX12
  - 5. To run the script, enter the command: addusers1.bat and press the Enter key.
  - 6. When prompted for the user password, enter: web1sphere  
(All users have the same password, which is based on this entry.)
  - 7. When the script completes and you are prompted, press **Enter** to exit the scripting interface.

```

Administrator: Command Prompt - addusers1.bat
Adding 7 users...
WASX7209I: Connected to process "server1" on node Node1 using SOAP connector; T
he type of process is: UnManagedProcess
uid=bpadminA,o=defaultWIMFileBasedRealm
uid=bpadminB,o=defaultWIMFileBasedRealm
uid=approverA,o=defaultWIMFileBasedRealm
uid=approverB,o=defaultWIMFileBasedRealm
uid=approverC,o=defaultWIMFileBasedRealm
uid=supervisorA,o=defaultWIMFileBasedRealm
uid=supervisorB,o=defaultWIMFileBasedRealm
Adding 3 groups and attaching users to groups...
WASX7209I: Connected to process "server1" on node Node1 using SOAP connector; T
he type of process is: UnManagedProcess
cn=bpadmins,o=defaultWIMFileBasedRealm
cn=approvers,o=defaultWIMFileBasedRealm
cn=supervisors,o=defaultWIMFileBasedRealm
uid=bpadminA,o=defaultWIMFileBasedRealm
uid=bpadminB,o=defaultWIMFileBasedRealm
uid=approverA,o=defaultWIMFileBasedRealm
uid=approverB,o=defaultWIMFileBasedRealm
uid=approverC,o=defaultWIMFileBasedRealm
uid=supervisorA,o=defaultWIMFileBasedRealm
uid=supervisorB,o=defaultWIMFileBasedRealm
"Attempting to refresh People Queries . . ."
WASX7209I: Connected to process "server1" on node Node1 using SOAP connector; T
he type of process is: UnManagedProcess
WASX7303I: The following options are passed to the scripting environment and are
available as arguments that are stored in the argv variable: "[-server, server1
]"
Refreshing all staff queries...
Done.
Press any key to continue . . .

```

- 8. Press any key when done and then close the command window.
- 9. Restart IBM Process Server by right-clicking **IBM Process Server v8.5.7 at localhost** and then clicking **Restart**.

- \_\_ 10. Examine the user information in the IBM Process Server administrative console.
- \_\_ a. Switch to the IBM Integration Designer. In the **Servers** view, right-click **IBM Process Server v8.5.7 at localhost** and choose **Administration > Run Administrative Console** from the menu.
  - \_\_ b. If you receive security alerts, click **Yes** to proceed.
  - \_\_ c. Use **User ID** `admin` and **Password** `web1sphere` and then click **Log in**.
  - \_\_ d. In the Navigator pane of the administrative console, expand **Users and Groups**.
  - \_\_ e. Click **Manage Users**.



## Troubleshooting

If after clicking **Manage Users** or **Manage Groups**, the web page is blank or does not list any users or groups, then run the `addusers1.bat` script and restart the IBM Process Server again.

- \_\_ f. Leave \* in the **Search for** field and click **Search**.
- \_\_ g. You see each of the users that were stated previously.

The screenshot shows the 'Manage Users' interface. At the top, there is a search bar labeled 'Search for Users' with dropdowns for 'Search by' (set to 'User ID') and 'Maximum results' (set to '100'). Below the search bar is a button labeled 'Search'. A message below the search bar states '17 users matched the search criteria.' Below this, there is a table with columns: 'Select', 'User ID', 'First name', 'Last name', 'E-mail', and 'Unique'. The table lists 17 users:

| Select                   | User ID                   | First name | Last name | E-mail                | Unique                   |
|--------------------------|---------------------------|------------|-----------|-----------------------|--------------------------|
| <input type="checkbox"/> | <a href="#">admin</a>     | admin      | admin     |                       | uid=admin,o=defaultW     |
| <input type="checkbox"/> | <a href="#">approverA</a> | approver   | A         | app_a@mycomp          | uid=approverA,o=defaultW |
| <input type="checkbox"/> | <a href="#">approverB</a> | approver   | B         | app_b@mycomp          | uid=approverB,o=defaultW |
| <input type="checkbox"/> | <a href="#">approverC</a> | approver   | C         | app_c@mycomp          | uid=approverC,o=defaultW |
| <input type="checkbox"/> | <a href="#">bdaniel</a>   | Bonnie     | Daniel    | Bonnie.Daniel@ibm.com | uid=bdaniel,o=defaultW   |
| <input type="checkbox"/> | <a href="#">bpadminA</a>  | bpadmin    | A         | adm_a@mycomp          | uid=bpadminA,o=defaultW  |

- h. You can repeat the same steps for the groups by clicking **Manage Groups** and searching for all group names (\*).

The screenshot shows the 'Manage Groups' interface. At the top, there is a search bar with 'Search by' set to 'Group name', 'Search for' containing an asterisk (\*), and 'Maximum results' set to 100. Below the search bar is a 'Search' button. A message below the search bar states '6 groups matched the search criteria.' The main area is a table titled 'Manage Groups' with columns: Select, Group name, Description, and Unique Name. The table lists six groups:

| Select                   | Group name                         | Description            | Unique Name                                      |
|--------------------------|------------------------------------|------------------------|--------------------------------------------------|
| <input type="checkbox"/> | <a href="#">Home Entertainment</a> | department 1 employees | cn=Home Entertainment,o=defaultWIMFileBasedRealm |
| <input type="checkbox"/> | <a href="#">Widget Division</a>    | department 2 employees | cn=Widget Division,o=defaultWIMFileBasedR        |
| <input type="checkbox"/> | <a href="#">approvers</a>          |                        | cn=approvers,o=defaultWIMFileBasedRealm          |
| <input type="checkbox"/> | <a href="#">bpadmins</a>           |                        | cn=bpadmins,o=defaultWIMFileBasedRealm           |
| <input type="checkbox"/> | <a href="#">managers</a>           | all managers           | cn=managers,o=defaultWIMFileBasedRealm           |
| <input type="checkbox"/> | <a href="#">supervisors</a>        |                        | cn=supervisors,o=defaultWIMFileBasedRealn        |

- i. Leave the administrative console open.

### Part 3: *Implementing security on the AccountVerification process module*

In this part of the exercise, you learn how to turn on global security and set up the security QoS on your service interface. You implement the security permission qualifier for the AccountVerification process to prevent access from the unauthorized users.

The purpose of this exercise is to introduce you to security settings, not to cover every possible security setting in detail. The settings that you configure in this exercise are emblematic of the types of settings you encounter in production applications.



#### Note

It is important to understand the difference between application security and people assignment criteria. Setting up application security defines who are the authorized users to start the SCA application. This feature ensures that no unauthorized users can access the application.

The purpose of the people assignment criteria is to identify sets of people that can be assigned to an instance-based authorization role. At run time, the people resolution uses the people assignment criteria to retrieve the user IDs and other user information. In this exercise, you

configure and test the application security through the AccountOpening user interface SCA client. You also configure and test instance-based authorization roles in inline and stand-alone human tasks to understand how instance-based authorization roles for the BPEL process works.

- \_\_\_ 1. Add the group **approvers** to the **Security permission** qualifier for the **InputCriterion** operation on the **AccountVerification** interface. This addition means that only members of the approvers group have access to start a process instance by starting the InputCriterion operation on the process interface. These steps are about enabling application security.
  - \_\_\_ a. In the **Projects** section of the **Business Integration** view, expand **FoundationModule** and double-click **Assembly Diagram**.
  - \_\_\_ b. Select the **AccountVerification** process component.
  - \_\_\_ c. Switch to the **Properties** view.
  - \_\_\_ d. Select the **All Qualifiers** tab.
  - \_\_\_ e. Click the **Security Qualifiers** tab to filter out all the other qualifiers for better readability.
  - \_\_\_ f. In the **Component** column, expand **AccountVerification > Interfaces > AccountVerification**.
  - \_\_\_ g. Click **InputCriterion**.

| Security                      |                     |                   |
|-------------------------------|---------------------|-------------------|
| Component                     | Security permission | Security identity |
| ▷ AccountVerificationExport   |                     |                   |
| ▲ AccountVerification         |                     |                   |
| Implementation                |                     |                   |
| Interfaces                    |                     |                   |
| ▲ AccountVerification         |                     |                   |
| InputCriterion                |                     |                   |
| References                    |                     |                   |
| CreditCheckServicePartner     |                     |                   |
| FinalApplicationReviewPartner |                     |                   |

- \_\_\_ h. Scroll to the right to the **Security** section.

- \_\_ i. Place the cursor in the **Security permission** column and enter: **approvers**

| Security                      |                     |                   |
|-------------------------------|---------------------|-------------------|
| Component                     | Security permission | Security identity |
| AccountVerificationExport     |                     |                   |
| AccountVerification           |                     |                   |
| Implementation                |                     |                   |
| Interfaces                    |                     |                   |
| AccountVerification           |                     |                   |
| InputCriterion                | approvers           |                   |
| References                    |                     |                   |
| CreditCheckServicePartner     |                     |                   |
| FinalApplicationReviewPartner |                     |                   |

- \_\_ j. Save your changes.
- \_\_ 2. Create a human task that identifies the user group “**approvers**” as the potential starters of the **AccountVerification** process.



### Note

In the previous step, you configured the application security where the security permission qualifier is set on the interface operation so that the unauthorized SCA clients cannot access the AccountVerification SCA module. In this step, you create people assignment criteria. To create people assignment criteria, you create the inline invocation task to identify those staff members who are allowed to send a request to the receive activity of the BPEL process. These staff members are called as potential starters in the inline invocation task.

- \_\_ a. Double-click **AccountVerification** in the assembly diagram to open the process in the BPEL editor.
- \_\_ b. In the AccountVerification process, select the **Account Verification Receive** activity.
- \_\_ c. In the **Properties** view, switch to the **Authorization** tab.
- \_\_ d. Click **New** to create a new potential starters task.
- \_\_ e. In the task editor, in the **People Assignment** section, select the **Potential Starters** role.
- \_\_ f. In the **Properties** view, for the **People assignment criteria** field, choose **Group Search** from the drop-down list.

- \_\_\_ g. In the table that is shown here, place the cursor in the **Value** column and enter approvers for the **GroupID**.

| Name         | Value     |
|--------------|-----------|
| GroupID      | approvers |
| Type         |           |
| IndustryType |           |
| BusinessType |           |

- \_\_\_ h. Save your changes and close the AccountVerification process.
- \_\_\_ 3. Add the group **supervisors** to the **Final Application Review** task. This addition means that only members of the supervisors group can approve riskier applications.
- \_\_\_ a. In the **Business Integration** view, expand **HumanTaskServices > Integration Logic > Human Tasks > htm**.
  - \_\_\_ b. Double-click **Final Application Review** to open it in the task editor.
  - \_\_\_ c. In the **Properties** view, switch to the **Details** tab.
  - \_\_\_ d. For **People directory**, choose **Virtual Member Manager** from the list.

|             |                         |                        |
|-------------|-------------------------|------------------------|
| Description | People directory:       | Virtual Member Manager |
| Details     | Task priority:          | 5                      |
| Propagation | Work basket identifier: |                        |
| Interface   | Business category:      |                        |
| Duration    |                         |                        |

- \_\_\_ e. If you receive a message that indicates the user registry is changed, click **OK**.
- \_\_\_ f. Save your changes.

- \_\_ g. In the **People Assignment** section, select the **Potential Owners** role.

The screenshot shows the 'To-do Task' editor for a task named 'FinalApplicationReview'. Under the 'Service Interface' section, there is a 'People Assignment (Receiver)' table. The 'Potential Creators' row is highlighted with a red border. Below it, the 'Potential Owners' row is also present. The 'User Interface' section shows an 'HTML-Dojo' entry.

|  | Potential Creators | Everybody |
|--|--------------------|-----------|
|  | Potential Owners   | Everybody |

- \_\_ h. In the **Properties** view, choose **Group Search** from the **People assignment criteria** list.  
 \_\_ i. In the table that is shown here, place the cursor in the **Value** column and enter supervisors for the **GroupID**.

The screenshot shows the 'Staff role - Potential Owners (Single ownership)' properties view. The 'Assign People' tab is selected. The 'People assignment criteria' dropdown is set to 'Group Search', which is highlighted with a red border. The 'Name' and 'Value' columns of the table below are also highlighted with a red border. The table contains three rows: GroupID (supervisors), Type, and IndustryType.

| Name         | Value       |
|--------------|-------------|
| GroupID      | supervisors |
| Type         |             |
| IndustryType |             |

- \_\_ j. Save your changes and close the task editor.

## Part 4: Enabling IBM Process Server security

To enforce the permissions for tasks, you must enable security on IBM Process Server. The steps necessary for enabling security on IBM Process Server are identical to the steps for enabling security on WebSphere Application Server. These steps are also identical for turning on security in the WebSphere test environment or a separate installation of IBM Process Server.

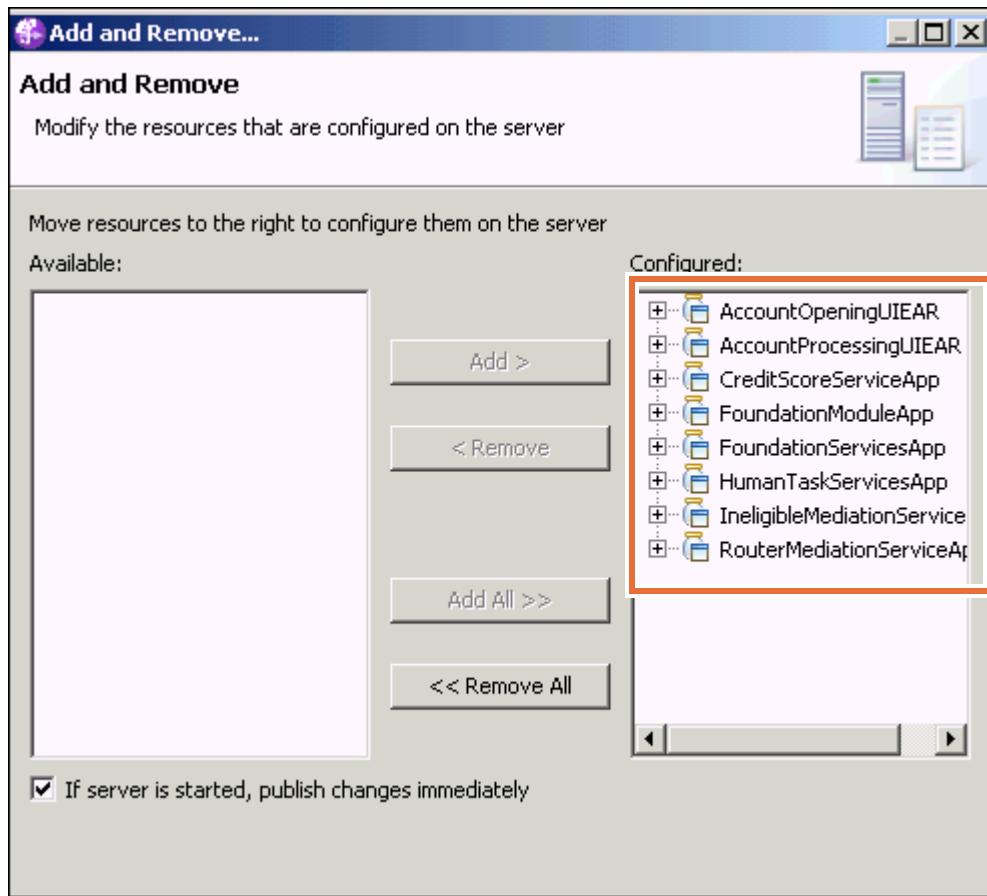
As part of enabling security, you must complete several steps. IBM Process Server can use the local operating system, an LDAP server, or a custom user registry for security. In this exercise, you

configure the server to use a WebSphere user directory (by selecting a federated repository) and specify a user ID and password that the server uses for checking authentication information. IBM Process Server supports only LTPA for an authentication mechanism; in this way, the SCA propagates the security information. Simple WebSphere Authentication Mechanism (SWAM) is not supported even though it is available on WebSphere Application Server. You must create the LTPA encryption key, which is used to establish the authentication token. You also must review the different server resources and verify that the correct authentication information is specified to run with security enabled.

- \_\_\_ 1. Open the IBM Process Server administrative console, if it is closed.
  - \_\_\_ a. In the **Servers** view, right-click **IBM Process Server v8.5.7 at localhost** and choose **Administration > Run Administrative Console** from the menu.
  - \_\_\_ b. Use **User ID** `admin` and **Password** `web1sphere` and then click **Log in**.
- \_\_\_ 2. Enable global security on the IBM Process Server instance.
  - \_\_\_ a. In the left Navigator, expand **Security**.
  - \_\_\_ b. Click **Global security**.
  - \_\_\_ c. In the **Authentication** section on the right, click the **LTPA** link.
  - \_\_\_ d. Under **Cross-cell single sign-on**, enter a value of `web1sphere` for the **Password** and **Confirm password** fields.

This value is used to generate the LTPA encryption key and is not required to be a valid password. The password is needed a second time only if the LTPA key is used on another system for implementing single sign-on capabilities.
  - \_\_\_ e. Click **OK**.
  - \_\_\_ f. Click the **Save** link at the top of the page to save the changes to the master configuration.
  - \_\_\_ g. Leave the administrative console open.
- \_\_\_ 3. Deploy all of the modules.
  - \_\_\_ a. In the **Servers** view, right-click **IBM Process Server v8.5.7 at localhost** and select **Add and Remove**.

- \_\_\_ b. Click **Add All** to add each of the projects to the **Configured** list.



- \_\_\_ c. Click **Finish**.
- \_\_\_ d. Wait until the modules are published and started. This process might take several minutes, depending on your system resources. To see the status of the modules, expand **IBM Process Server v8.5.7 at localhost** in the **Servers** view.
- \_\_\_ 4. Map users to roles.

You assigned the roles to the **AccountVerification** business process in previous steps. Those assignments were role names, not the actual user or group names. For security to work, all of the applications that contain roles must map the roles to a user or group.

- \_\_\_ a. Switch to the IBM Process Server administrative console.
- \_\_\_ b. Click **Applications > Application Types > WebSphere enterprise applications**.
- \_\_\_ c. Scroll down until you find **FoundationModuleApp**. Click **FoundationModuleApp**.

|                          |                                                           |  |
|--------------------------|-----------------------------------------------------------|--|
| <input type="checkbox"/> | <a href="#">CreditScoreServicesApp</a>                    |  |
| <input type="checkbox"/> | <a href="#">DefaultApplication</a>                        |  |
| <input type="checkbox"/> | <a href="#">FoundationModuleApp</a>                       |  |
| <input type="checkbox"/> | <a href="#">FoundationServicesApp</a>                     |  |
| <input type="checkbox"/> | <a href="#">HTM_PredefinedTaskMsg_V8000_Node1_server1</a> |  |

- \_\_ d. Click **Security role to users/group mapping**. This link is visible only for the applications that contain roles.

The screenshot shows the 'Enterprise Applications' configuration page for the 'FoundationModuleApp'. The 'General Properties' section includes fields for 'Name' (set to 'FoundationModuleApp') and 'Application reference validation' (set to 'Issue warnings'). The 'Detail Properties' section lists several options, with 'Security role to user/group mapping' highlighted by a red box. Other listed items include 'Target specific application status', 'Startup behavior', 'Application binaries', 'Class loading and update detection', 'Request dispatcher properties', 'JASPI provider', 'Custom properties', 'View Deployment Descriptor', and 'Last participant support extension'. To the right, there are sections for 'Modules' (with links to 'Manage Modules' and 'Display module build ids'), 'Web Module Properties' (with links to 'Session management', 'Context Root For Web Modules', 'JSP and JSF options', and 'Virtual hosts'), 'Enterprise Java Bean Properties' (with a link to 'Default messaging provider references'), 'Client Module Properties' (with a link to 'Client module deployment mode'), and 'Web Services Properties'.

- \_\_ e. Notice that currently no users or groups are mapped to the approvers role. Check the **approvers** role and click **Map Groups**.

**Enterprise Applications**

[Enterprise Applications](#) > [FoundationModuleApp](#) > Security role to user/group mapping

Security role to user/group mapping

Each role that is defined in the application or module must map to a user or group from the domain us accessIds: The accessIds are required only when using cross realm communication in a multi domain se all other scenarios the accessId will be determined during the application start based on the user or gro The accessIds represent the user and group information that is used for Java Platform, Enterprise Editic authorization when using the WebSphere default authorization engine. The format for the accessIds is user:realm/uniqueUserID, group:realm/uniqueGroupID. Entering wrong information in these fields will c authorization to fail. AllAuthenticatedInTrustedRealms: This indicates that any valid user in the trusted given the access. AllAuthenticated: This indicates that any valid user in the current realm be given the a

| Select                              | Role      | Special subjects | Mapped users | Mapped groups |
|-------------------------------------|-----------|------------------|--------------|---------------|
| <input checked="" type="checkbox"/> | approvers | None             |              |               |

OK Cancel

- \_\_ f. Click **Search**.

- g. Select **approvers** from the **Available** group list, and click the right arrow to select the **approvers** group.

The screenshot shows the 'Enterprise Applications' interface with the following details:

- Enterprise Applications > FoundationModuleApp > Security role to user/group mapping > Map user**
- Available:** Home Entertainment, bpadmins, **approvers**, Widget Division, managers
- Selected:** (empty)
- Buttons:** Search string (\*), Search, Right arrow (highlighted with a red box), Left arrow

- h. Notice that the **approvers** group is listed in the **Selected** group list.

The screenshot shows the 'Enterprise Applications' interface with the path: Enterprise Applications > FoundationModuleApp > Security role to user/group mapping > Map users. A message at the top says: 'Use this page to search for users or groups and add them to the selected roles.' Below it, a section titled 'Search and Select Groups' has instructions: 'Decide how many results to display, enter a search string (use \* for wildcard), and click Search. See Available list and add them to the Mapped to role list.' It includes fields for 'Display a maximum of' (set to 20) and 'Search string' (\*). A 'Search' button is present. On the left, a 'Available:' list contains: Home Entertainment, bpadmins, supervisors, Widget Division, managers. On the right, a 'Selected:' list contains: approvers (highlighted with a red box). Two blue circular arrows are positioned between the two lists.

- i. Click **OK**. You are back on the **Security role to users/group mapping** page. Notice that now the **approvers** role is mapped to the **approvers** group.

The screenshot shows the 'Enterprise Applications' interface with the path: Enterprise Applications > FoundationModuleApp > Security role to user/group mapping. A message at the top says: 'Each role that is defined in the application or module must map to a user or group from the domain user accessIDs: The accessIDs are required only when using cross realm communication in a multi domain scenario. In all other scenarios the accessId will be determined during the application start based on the user or group information. The accessIDs represent the user and group information that is used for Java Platform, Enterprise Edition (J2EE) authorization when using the WebSphere default authorization engine. The format for the accessIDs is user:realm/uniqueUserID, group:realm/uniqueGroupID. Entering wrong information in these fields will cause authorization to fail. AllAuthenticatedInTrustedRealms: This indicates that any valid user in the trusted realms be given the access. AllAuthenticated: This indicates that any valid user in the current realm be given the access.' Below this, there are three buttons: 'Map Users...', 'Map Groups...', and 'Map Special Subjects...'. A table below shows the mapping: Role 'approvers' is mapped to 'approvers' in the 'Mapped groups' column. Buttons 'OK' and 'Cancel' are at the bottom.

| Select                   | Role      | Special subjects | Mapped users | Mapped groups |
|--------------------------|-----------|------------------|--------------|---------------|
| <input type="checkbox"/> | approvers | None             |              | approvers     |

- \_\_\_ j. Click **OK**.
- \_\_\_ k. You are back on the **FoundationModuleApp** page. Click **Save** to save your changes.

## Part 5: Testing security

- \_\_\_ 1. Try to create an instance of the **AccountVerification** process through the **AccountOpeningUI** human task user interface. Log in to the test client with the **superVisorA** user. Since this user is not from the approvers role for which security permission QoS was defined, it must not allow the user to create an instance.
  - \_\_\_ a. Open a Firefox browser window.
  - \_\_\_ b. Enter the address:  
`https://localhost:9443/AccountOpeningUI/Login.jsp`
  - \_\_\_ c. Enter **superVisorA** (note the case) in the **Name** field and **web1sphere** in the **Password** field, and click **Login**.
  - \_\_\_ d. In the **Business Case** section of the Business User Client page, click **New**.
  - \_\_\_ e. In the **Task** section of the **Business Cases > New** page, click **CreateApplication**.
  - \_\_\_ f. In the **Input Data** section of the **Business Cases > New > CreateApplication** page, enter **ACME123** in the **accountNumber** field and **ACME** in the **companyName** field. Leave the other fields blank and click **Create** at the bottom of the page.
  - \_\_\_ g. Switch to the **Servers** view in IBM Integration Designer.
  - \_\_\_ h. Right-click **IBM Process Server V8.5.7 at localhost** and choose **Launch > Business Process Choreographer Explorer** from the menu.
  - \_\_\_ i. If a security alerts window prompts you, click **Yes** to proceed.
  - \_\_\_ j. In the Business Process Choreographer Explorer login page, enter **superVisorA** in the **User Name** field and **web1sphere** in the **Password** field, and click **Login**.

- \_\_ k. In the **Task Instances** section, click **All Tasks**.

The screenshot shows the 'Views' tab selected in the top navigation bar. The left sidebar contains several sections: 'Process Templates' (with 'Currently Valid' and 'All Versions' links), 'Process Instances' (with 'Started By Me', 'Administered By Me', 'Critical Processes', 'Terminated Processes', and 'Failed Compensations' links), 'Activity Instances' (with 'Stopped Activities' link), 'Task Templates' (with 'My Task Templates' link), and 'Task Instances'. The 'Task Instances' section is expanded, showing 'My To-dos', 'All Tasks' (which is highlighted with a red box), 'Initiated By Me', 'Administered By Me', and 'My Escalations'.

- \_\_ l. In the All Tasks view, notice that the CreateApplication task is in the Failed state. To view the cause of failure, click the **CreateApplication** link.

The screenshot shows the 'All Tasks' view with a single task instance listed. The table has columns: Priority, Task Name, State, Kind, Owner, Originator, and Escalation. The data row shows: Priority 5, Task Name 'CreateApplication' (highlighted with a red box), State 'Failed' (highlighted with a red box), Kind 'Invocation Task', Owner 'supervisorA', Originator 'supervisorA', and Escalation 'no'. Below the table, it says 'Items found: 1 Items selected: 0' and shows page navigation.

|                          | Priority | Task Name         | State  | Kind            | Owner       | Originator  | Escalation |
|--------------------------|----------|-------------------|--------|-----------------|-------------|-------------|------------|
| <input type="checkbox"/> | 5        | CreateApplication | Failed | Invocation Task | supervisorA | supervisorA | no         |

m. In the Task Instance page, click the **Task Fault Message** tab and click **View Source**.

The screenshot shows the 'Task Instance' page. At the top, there are four buttons: Delete, Restart, Reschedule, and Change Business Category. Below them is a section titled 'Task Description' with fields for Task Name (CreateApplication), Description, and Reason (Starter, Administrator, Originator). A navigation bar below contains tabs: Details, Template Details, Task Input Message, Task Output Message, and Task Fault Message. The 'Task Fault Message' tab is highlighted with a red box. Below this is a 'Form View' section containing two tables. The first table has columns for exceptionMessage and locale, with rows for default (CWTKE0010I: A run-time exception was received for the Service and the invocation task (originating task)) and en\_US (CWTKE0010I: A run-time exception was received for the Service and the invocation task (originating task)). The second table has columns for exceptionStackTrace and message, with a single row showing CWTKE0010I: A run-time exception was received for the Service Component Architecture (SCA) service. At the bottom left of the 'Form View' section is a 'View Source' button, which is also highlighted with a red box.

| exceptionMessage | locale  | message                                                                                                   |
|------------------|---------|-----------------------------------------------------------------------------------------------------------|
|                  | default | CWTKE0010I: A run-time exception was received for the Service and the invocation task (originating task). |
|                  | en_US   | CWTKE0010I: A run-time exception was received for the Service and the invocation task (originating task). |

| exceptionStackTrace | message                                                                                            |
|---------------------|----------------------------------------------------------------------------------------------------|
|                     | CWTKE0010I: A run-time exception was received for the Service Component Architecture (SCA) service |

- \_\_\_ n. In the Source View, scroll down and notice that an exception is thrown.

The screenshot shows the 'Source View' tab of a software interface. At the top, there are tabs: Details, Template Details, Task Input Message, Task Output Message, and Task Fault Message. The Task Fault Message tab is selected. Below the tabs, the title 'Source View' is displayed. The main area contains XML code. A red rectangular box highlights the following portion of the code:

```

<exceptionMessage>
 <message>CWTKE0010I: A run-time exception was received for the Service Component Architecture (SCA) service and the invocation task (originating task).</message>
</exceptionMessage>
<exceptionMessage>
 <locale>en_US</locale>
 <message>CWTKE0010I: A run-time exception was received for the Service Component Architecture (SCA) service and the invocation task (originating task).</message>
</exceptionMessage>
<exceptionStackTrace>CWTKE0010I: A run-time exception was received for the Service Component Architecture (SCA) service and the invocation task (originating task).
com.ibm.websphere.sca.ServiceRuntimeException: Permission denied: User supervisorA is not in role:approvers and does not have permission to invoke the method
com.ibm.ws.sca.internal.security.handler.SecurityPermissionNativeHandler.process

```

Below the code area, there is a 'Back' button.

- \_\_\_ o. Click **Delete** at the top of the page to delete this failed instance.
- \_\_\_ p. Log out of the Business Process Choreographer Explorer but *do not* close it.
- \_\_\_ 2. Try to create instance of the **AccountVerification** process through the AccountOpeningUI human task user interface. Log in to the test client with the approverA user. Since this user is from the approvers role for which security permission QoS was defined, it must be allowed to create an instance.
- \_\_\_ a. Switch to the AccountOpeningUI human task user interface and log out. You are logging out so that you can log in with the other user name. If the user interface was closed, open the Firefox browser and enter the address to reopen the user interface:  
<https://localhost:9443/AccountOpeningUI/Login.jsp>  
The Login to Business User Client web page is opened.
- \_\_\_ b. Enter approverA in the **Name** field and web1sphere in the **Password** field, and click **Login**.
- \_\_\_ c. In the **Business Case** section of the Business User Client page, click **New**.
- \_\_\_ d. In the **Task** section of the **Business Cases > New** page, click **CreateApplication**.
- \_\_\_ e. In the **Input Data** section of the **Business Cases > New > CreateApplication** page, enter ACME123 in the **accountNumber** field and ACME in the **companyName** field. Leave the other fields blank and click **Create** at the bottom of the page.
- \_\_\_ f. Switch to the **Servers** view in IBM Integration Designer.
- \_\_\_ g. Switch to the Business Process Choreographer Explorer login page. Enter approverA in the **User Name** field and web1sphere in the **Password** field, and click **Login**.

- \_\_ h. In the **Task Instances** section, click **All Tasks**.
- \_\_ i. In the **All Tasks** view now, you see successful creation of the **CreateApplication** instance and the **Account Verification Receive** human task.

**All Tasks**

Use this page to work with task instances for which you have access rights. [\[i\]](#)

|                          | Priority | Task Name                    | State   | Kind            | Owner     | Originator | Escalation |
|--------------------------|----------|------------------------------|---------|-----------------|-----------|------------|------------|
| <input type="checkbox"/> | 5        | Account Verification Receive | Running | Invocation Task | approverA | no         |            |
| <input type="checkbox"/> | 5        | CreateApplication            | Running | Invocation Task | approverA | no         |            |

Items found: 2 Items selected: 0 << Page 1 of 1 >> Items per page: 20

- \_\_ j. In the **Process Instances** section, click **Administered By Me**, select the running process instance, and click **Terminate**.

**Process Instances Administered By Me**

Use this page to work with process instances for which you are the process administrator. [\[i\]](#)

| Migrate                                                                    | Terminate                           | Delete                 | Suspend                | Resume                 | Restart                | Compensate             | Claim Ownership        |
|----------------------------------------------------------------------------|-------------------------------------|------------------------|------------------------|------------------------|------------------------|------------------------|------------------------|
| <input type="checkbox"/>                                                   | <input checked="" type="checkbox"/> | <input type="button"/> |
| Process Instance Name                                                      |                                     | Process Template Name  |                        |                        |                        |                        |                        |
| <input checked="" type="checkbox"/> PI:90030159.9a2950f7.a8afef53.32bd00c8 |                                     | AccountVerification    |                        |                        |                        |                        |                        |

Items found: 1 Items selected: 1 << Page 1 of 1 >> Item

- \_\_ k. Again, select the running process instance and click **Delete**.
- \_\_ l. Click **Logout** at the top of the page to log out approverA from the Business Process Choreographer.
- \_\_ m. Test the Receive authorization inline human task. This test is to confirm that only authorized users can invoke the Receive activity of the BPEL process. Try to start the instance with superVisorA user. As this user is not authorized to send a request to a Receive activity of the BPEL process, the instance must not start.
  - \_\_ a. In the Business Process Choreographer Explorer login page, log in as superVisorA. Recall that you set the approvers group as the potential starters for this process and the superVisorA is from the superVisors group, so the instance must fail to start.

- \_\_ b. In the **Process Templates** section, click **Currently Valid**. The **Currently Valid Process Templates** view opens.

| Process Template Name                                   | Valid From               | Process App |
|---------------------------------------------------------|--------------------------|-------------|
| <input checked="" type="checkbox"/> AccountVerification | 9/15/2014 3:38:09 AM PDT |             |

Items found: 1 Items selected: 1

- \_\_ c. Check the **AccountVerification** process template and then click **Start Instance**.

| Process Template Name                                   | Valid From               | Process App |
|---------------------------------------------------------|--------------------------|-------------|
| <input checked="" type="checkbox"/> AccountVerification | 9/15/2014 3:38:09 AM PDT |             |

Items found: 1 Items selected: 1

- \_\_ d. Notice that the exception message is opened. This exception is because the logged-in user superVisorA is not the authorized user to start the process application.

| Process Template Name                                   | Valid From               | Process App | Snapshot | Long Running |
|---------------------------------------------------------|--------------------------|-------------|----------|--------------|
| <input checked="" type="checkbox"/> AccountVerification | 9/15/2014 3:38:09 AM PDT |             |          | yes          |

Items found: 1 Items selected: 1

- \_\_ 4. Try to start the **AccountVerification** process with the sufficient privileged user.

- \_\_ a. Log out the existing user (superVisorA) from the Business Process Choreographer Explorer and log in with approverA as the user ID.

- \_\_ b. In the Business Process Choreographer Explorer login page, enter approverA in the **User Name** field and web1sphere in the **Password** field, and click **Login**.
- \_\_ c. In the **Process Templates** section, click **Currently Valid**. The **Currently Valid Process Templates** view is opened.
- \_\_ d. Check the **AccountVerification** process template and then click **Start Instance**. Notice that since the current user (approverA) has sufficient privilege to start the process instance, the new instance is created, and you do not get an exception message.
- \_\_ e. In the **Process Input Message** page, enter the following fields; leave rest of the fields blank, and then click **Submit**.
  - Process Name: TestA
  - accountNumber: ACM002
  - companyName: ACME

**Process Input Message**

Use this page to provide the input that is needed to start an instance of a process. [\[i\]](#)

**Submit**

| Process Template Name | AccountVerification                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |            |  |       |  |               |                                     |                 |                      |                     |                       |          |                      |             |                                   |                  |                      |                 |                      |                    |                      |              |                      |
|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------|--|-------|--|---------------|-------------------------------------|-----------------|----------------------|---------------------|-----------------------|----------|----------------------|-------------|-----------------------------------|------------------|----------------------|-----------------|----------------------|--------------------|----------------------|--------------|----------------------|
| Process Description   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |            |  |       |  |               |                                     |                 |                      |                     |                       |          |                      |             |                                   |                  |                      |                 |                      |                    |                      |              |                      |
| Operation             | InputCriterion                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |            |  |       |  |               |                                     |                 |                      |                     |                       |          |                      |             |                                   |                  |                      |                 |                      |                    |                      |              |                      |
| Process Name          | <input type="text" value="TestA"/>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |            |  |       |  |               |                                     |                 |                      |                     |                       |          |                      |             |                                   |                  |                      |                 |                      |                    |                      |              |                      |
| Process Input Message | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th colspan="2" style="text-align: left;">Form View*</th> </tr> <tr> <th style="width: 15%;">Input</th> <th></th> </tr> </thead> <tbody> <tr> <td>accountNumber</td> <td><input type="text" value="ACM002"/></td> </tr> <tr> <td>applicationDate</td> <td><input type="text"/></td> </tr> <tr> <td>applicationDecision</td> <td>- <a href="#">Add</a></td> </tr> <tr> <td>comments</td> <td><input type="text"/></td> </tr> <tr> <td>companyName</td> <td><input type="text" value="ACME"/></td> </tr> <tr> <td>contactFirstName</td> <td><input type="text"/></td> </tr> <tr> <td>contactLastName</td> <td><input type="text"/></td> </tr> <tr> <td>contactPhoneNumber</td> <td><input type="text"/></td> </tr> <tr> <td>creditRating</td> <td><input type="text"/></td> </tr> </tbody> </table> | Form View* |  | Input |  | accountNumber | <input type="text" value="ACM002"/> | applicationDate | <input type="text"/> | applicationDecision | - <a href="#">Add</a> | comments | <input type="text"/> | companyName | <input type="text" value="ACME"/> | contactFirstName | <input type="text"/> | contactLastName | <input type="text"/> | contactPhoneNumber | <input type="text"/> | creditRating | <input type="text"/> |
| Form View*            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |            |  |       |  |               |                                     |                 |                      |                     |                       |          |                      |             |                                   |                  |                      |                 |                      |                    |                      |              |                      |
| Input                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |            |  |       |  |               |                                     |                 |                      |                     |                       |          |                      |             |                                   |                  |                      |                 |                      |                    |                      |              |                      |
| accountNumber         | <input type="text" value="ACM002"/>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |            |  |       |  |               |                                     |                 |                      |                     |                       |          |                      |             |                                   |                  |                      |                 |                      |                    |                      |              |                      |
| applicationDate       | <input type="text"/>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |            |  |       |  |               |                                     |                 |                      |                     |                       |          |                      |             |                                   |                  |                      |                 |                      |                    |                      |              |                      |
| applicationDecision   | - <a href="#">Add</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |            |  |       |  |               |                                     |                 |                      |                     |                       |          |                      |             |                                   |                  |                      |                 |                      |                    |                      |              |                      |
| comments              | <input type="text"/>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |            |  |       |  |               |                                     |                 |                      |                     |                       |          |                      |             |                                   |                  |                      |                 |                      |                    |                      |              |                      |
| companyName           | <input type="text" value="ACME"/>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |            |  |       |  |               |                                     |                 |                      |                     |                       |          |                      |             |                                   |                  |                      |                 |                      |                    |                      |              |                      |
| contactFirstName      | <input type="text"/>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |            |  |       |  |               |                                     |                 |                      |                     |                       |          |                      |             |                                   |                  |                      |                 |                      |                    |                      |              |                      |
| contactLastName       | <input type="text"/>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |            |  |       |  |               |                                     |                 |                      |                     |                       |          |                      |             |                                   |                  |                      |                 |                      |                    |                      |              |                      |
| contactPhoneNumber    | <input type="text"/>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |            |  |       |  |               |                                     |                 |                      |                     |                       |          |                      |             |                                   |                  |                      |                 |                      |                    |                      |              |                      |
| creditRating          | <input type="text"/>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |            |  |       |  |               |                                     |                 |                      |                     |                       |          |                      |             |                                   |                  |                      |                 |                      |                    |                      |              |                      |

- \_\_ 5. Try to claim the **FinalApplicationReview** human task.
  - \_\_ a. Click **My To-dos** underneath **Task Instances** and notice that no task is available to claim for the current logged-in user (approverA). Recall that the **FinalApplicationReview** human task is configured for the superVisors group as the potential owner of the task.

- \_\_ b. Log out the existing user (approverA) from the Business Process Choreographer Explorer and log in with user ID: superVisorA
- \_\_ c. In the Business Process Choreographer Explorer login page, enter superVisorA in the **User Name** field and web1sphere in the **Password** field, and click **Login**.
- \_\_ d. When you log in as superVisorA, the **My To-dos** page is opened. As the current logged-in user (superVisorA) is a privileged user, you see two **FinalApplicationReview** tasks ready to claim. Click any one check box for the **Final Application Review** task to test and click **Work On**.

| Priority | Task Name                | State | Kind       | Owner     | Originator |
|----------|--------------------------|-------|------------|-----------|------------|
| 5        | Final Application Review | Ready | To-do Task | approverA |            |
| 5        | Final Application Review | Ready | To-do Task | approverA |            |

Items found: 2 Items selected: 0 << Page 1 of 1 >> Items

- \_\_ e. At the top of the page, click **Complete**. This test shows that the user with sufficient privilege (superVisorA) was able to claim and complete the task that is assigned.
- \_\_ f. Log out of the Business Process Choreographer Explorer.
- \_\_ 6. Close all open editors.
- \_\_ 7. Close all open browsers.
- \_\_ 8. Right-click **IBM Process Server v8.5.7 at localhost**.
- \_\_ 9. Choose **Add and Remove** from the menu.
- \_\_ 10. Click **Remove All** and click **Finish**.
- \_\_ 11. (Optional) Stop IBM Process Server. IBM Process Center Server is required to test the next part of this exercise, not IBM Process Server. You might also leave IBM Process Server running throughout this exercise; but be aware that running two WebSphere Application Server profiles concurrently places a tremendous strain on system resources, particularly in a virtual environment. If you choose to leave the IBM Process Server running, expect a severe impact on performance.
  - \_\_ a. To stop IBM Process Server, in the **Servers** view, right-click **IBM Process Server v8.5.7 at localhost**.
  - \_\_ b. Choose **Stop** from the menu.
- \_\_ 12. Close the IBM Integration Designer. Click **File > Exit**, or click **X** in the upper-right corner.

# Securing activities of a process application in Process Designer

In this portion of the exercise, you experience how to create users and groups, and assign them to a participant group. You further learn how to configure a participant lane to use the created participant group. With this configuration, you enable security to all the activities in the participant lane so that only users of that participant group can participate in the activities of that lane in the business process definition. To enable security in IBM Process Center, you have two options to manage users, by using internal security providers or by using external security providers (like LDAP). In this exercise, you use the internal security provider, a federated repository.

## ***Part 1: Start the Process Center environment***

- \_\_\_ 1. Start the Process Center deployment manager.
  - \_\_\_ a. On your Windows desktop, select the shortcut that is titled: **Start Process Center deployment manager**. Double-click or press Enter to open the shortcut. It takes several minutes for deployment manager to start. When the deployment manager starts, you are prompted to press any key to continue. Press any key to close the command window.
- \_\_\_ 2. Start the Process Center node agent.
  - \_\_\_ a. On your Windows desktop, select the shortcut that is titled: **Start Process Center node agent**. Double-click or press Enter to open the shortcut. It takes several minutes for the node agent to start. When the node agent starts, you are prompted to press any key to continue. Press any key to close the command window.
- \_\_\_ 3. Start the Process Center cluster.
  - \_\_\_ a. On your Windows desktop, select the shortcut that is titled: **Start Process Center Cluster**. Double-click or press Enter to open the shortcut. It takes several minutes (longer than deployment manager and node agent processes) for the cluster to start. When the cluster starts, you are prompted to press any key to continue. Press any key to close the command window.

## ***Part 2: Creating users and groups***

- \_\_\_ 1. Create a user.
  - \_\_\_ a. Open Firefox browser and enter the following address to open the process admin console: `http://localhost:9081/ProcessAdmin/login.jsp`
  - \_\_\_ b. If prompted with security exceptions, then add the exception as needed.

- \_\_ c. Use **User Name** pcdeadmin and **Password** web1sphere and then click **Login**.

The image shows the IBM Process Admin Console login interface. At the top left is the IBM logo. To its right, the title "Process Admin Console" is displayed. Below this is a large input field labeled "User name" containing the text "pcdeadmin". Underneath it is another input field labeled "Password" containing a series of dots representing the password. At the bottom of the form is a blue "Log In" button. At the very bottom of the screen, there is a footer note: "Licensed Materials - Property of IBM. © Copyright IBM Corporation 2000, 2016. All Rights Reserved. US Government Users Restricted Rights- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp."

- \_\_ d. To create users, in the Navigator pane of the console, expand **User Management** and click **User Management**.
- \_\_ e. In the **Maintain User Settings** page in the **Internal IBM BPM User Details** section, enter **AM\_user1** as **User Name**, **Account Management User1** as **Full Name**, and **web1sphere** as **Password**. Enter **web1sphere** as **Confirm Password**. Click **Add** at the bottom.

The image shows the "User Management > Maintain User Settings" page. On the left, there is a "Retrieve Profile" section with a search bar and a "Retrieve" button. To the right of this is a "File Registry Users" list area. Further right is a "User Details" form, which is highlighted with a red border. This form contains four input fields: "User Name" with the value "AM\_user1", "Full Name" with the value "Account Management User1", "Password" with a series of dots, and "Confirm Password" with a series of dots. The entire "User Details" section is enclosed in a red rectangular border.

- \_\_\_ f. Repeat the previous steps to create a user with the following attributes:
- User Name: AM\_user2
  - Password: web1sphere
  - Full Name: Account Management User2
- \_\_\_ 2. Create a group.
- \_\_\_ a. In the Navigator pane of the console, expand **User Management** and click **Group Management**.
- \_\_\_ b. In the **Group Management** page, click **New Group**.

The screenshot shows the IBM Process Admin Console interface. The top navigation bar includes tabs for 'Server Admin' (which is selected), 'Process Inspector', and 'Installed Apps'. On the left, a 'Navigator' pane lists categories like 'IBM BPM Admin', 'User Management' (expanded to show 'User Management', 'Group Management', 'Bulk User Attribute Assignment', and 'User Synchronization'), 'Monitoring', 'Event Manager', and 'Admin Tools'. The main content area is titled 'User Management > Group Management'. It features a search bar 'Select Group to Modify:' and a 'New Group' button, which is highlighted with a red box. To the right of the 'New Group' button is a 'Remove' link and a scroll bar.

- \_\_\_ c. In the **Create Group** window, enter **AM\_group** as **Name** and **Account Management Group** as **Description**, and click **Save**.



— 3. Add members to the group.

- a. In the **Group Management** page, enter **AM** in the **Select Group to Modify** field to list all the existing groups with the initial **AM**. Select the **AM\_group**.

The screenshot shows the 'User Management > Group Management' page. The left sidebar has a tree view with 'User Management' expanded, showing 'User Management', 'Group Management' (selected), 'Monitoring', 'Event Manager', and 'Admin Tools'. The main panel has a title 'User Management > Group Management' and a search bar 'Select Group to Modify' containing 'AM'. Below it is a table with one row: 'New Group' (button) and 'AM\_group' (highlighted with a red border). To the right of the table are 'Remove' and 'Edit' buttons.

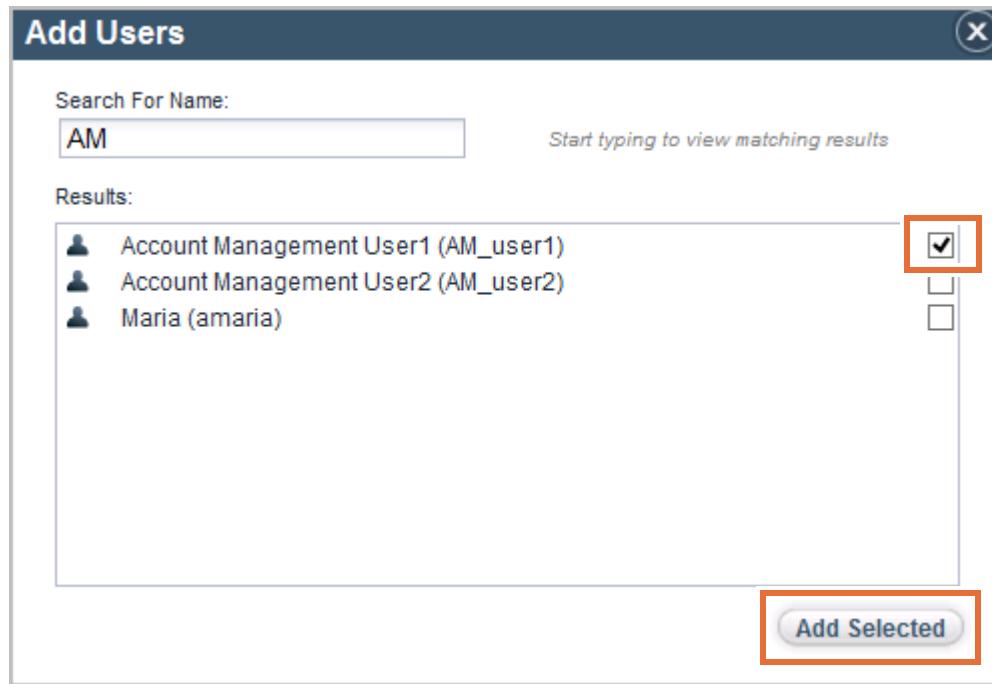
- b. Click **Add Users**.

The screenshot shows the 'Add Users' window. It has a search bar 'Search For Name' with 'AM' entered, and a note 'Start typing to view matching results'. Below it is a table titled 'AM\_group' with one row: 'New Group' (button) and 'AM\_group' (highlighted with a green background). To the right are 'Remove' and 'Edit' buttons. On the right side, there is a section for 'Team Manager Group (deprecated)' with 'No Team Manager Group' and buttons for 'Add Users' (highlighted with a red border) and 'Add Groups'.

- c. In the **Add User** window, enter **AM** in the **Search For Name** field to list all the users with **AM** initials.

The screenshot shows the 'Add User' window. It has a search bar 'Search For Name' with 'AM' entered. Below it is a table titled 'Results' with three rows: 'Account Management User1 (AM\_user1)', 'Account Management User2 (AM\_user2)', and 'Maria (amaria)'. Each row has a small user icon and a checkbox column to its right.

- \_\_ d. Check the **Account Management User1(AM\_user1)** user and click **Add Selected**. Notice that you are *not* adding the user AM\_user2 to the group.



- \_\_ e. Verify that the user is added to the group.

- \_\_ f. Click the **Logout** link available at the upper right of the page to log out of the Process Admin Console, and close the browser.

### **Part 3: Assigning a participant group to the lanes**

In this part of the exercise, you import the already created process application and then define a participant group in its business process definition (BPD). Then, in the BPD you assign the participant group to the lane. Since the purpose of this part of the exercise is to understand assigning security to the lane activities, you are not developing the process application from start. The purpose of assigning the participant group is to define the users who can participate in the activities inside the lane.

- \_\_ 1. Start the IBM Process Designer 8.5.7.
- \_\_ a. On the Windows desktop, locate the icon that is labeled **Start IBM Process Designer**.

- \_\_ b. Double-click the icon or press **Enter** to start IBM Process Designer.
  - \_\_ c. As the splash screen for IBM Process Designer loads, you are prompted to enter a user name and password to connect to the IBM Process Center repository. Use `pcdeadmin` as **User name** and `websphere` as **Password**.
  - \_\_ d. Click **Log In**.
  - \_\_ e. If you receive security alerts, click **Yes** to proceed. After a few moments, IBM Process Designer starts.
  - \_\_ f. If the Getting Started with IBM Process Designer 8.5.7.0 panel is displayed, then close it by clicking the cross mark on the upper-right corner of the window.
- \_\_ 2. Work with the process application.
- \_\_ a. If you are not in the Process Center perspective, click the **Process Center** link from the upper-right corner to go to the Process Center perspective.

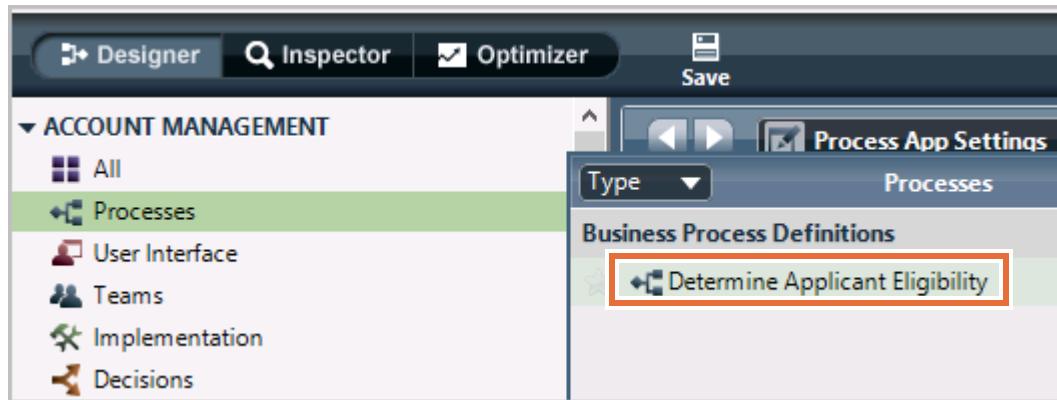


- \_\_ b. Make sure that you are in the **Process Apps** tab.
- \_\_ c. In the **Process Apps** tab, locate the process app named **Account Management (AM\_1012)**.
- \_\_ d. Click the **Open in Designer** link to that is associated with the process app. The process application opens in Designer view.

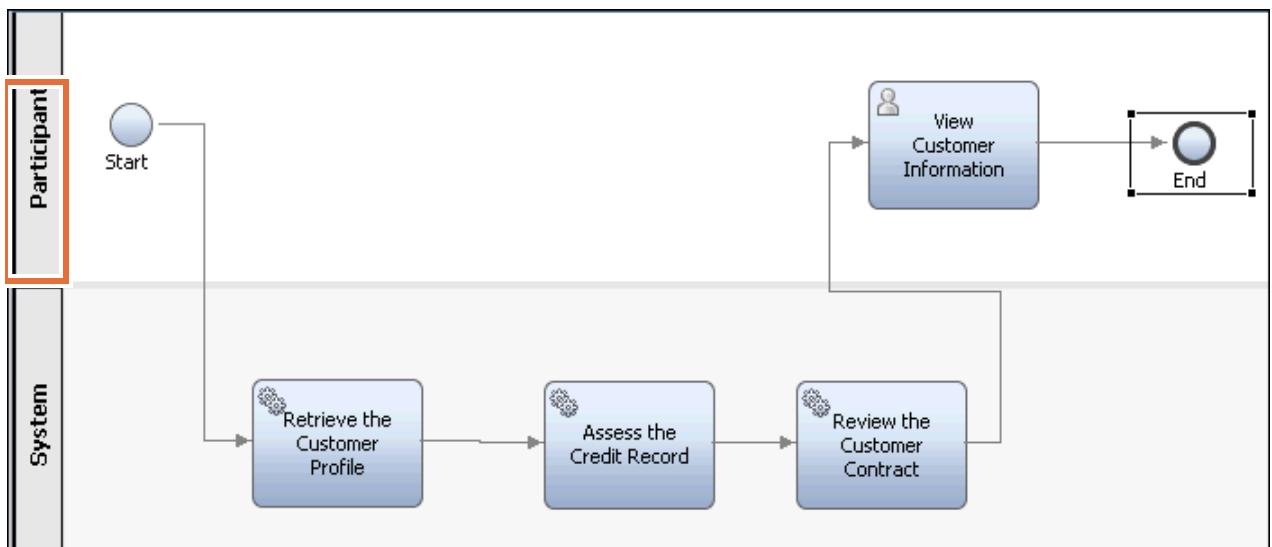
**Note**

Make sure that you click **Open in Designer** against **Account Management** with Acronym **AM\_1012** (not AM\_101) for this lab.

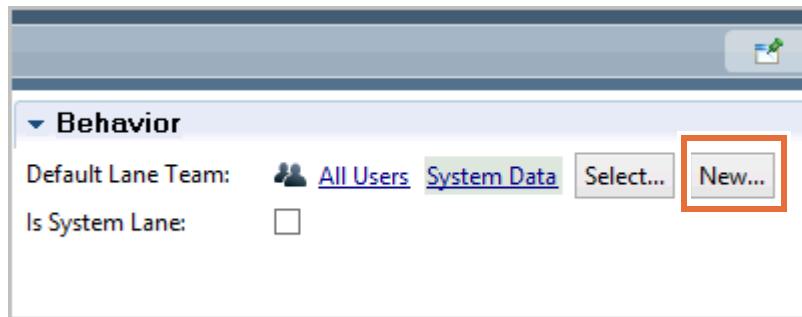
- \_\_\_ 3. Create the participant group.
  - \_\_\_ a. In the left pane of the design window, under the **Account Management** header, select **Processes**. It shows all the Business Process Definitions (BPD) that are available. Double-click **Determine Applicant Eligibility** BPD to open it in an editor.



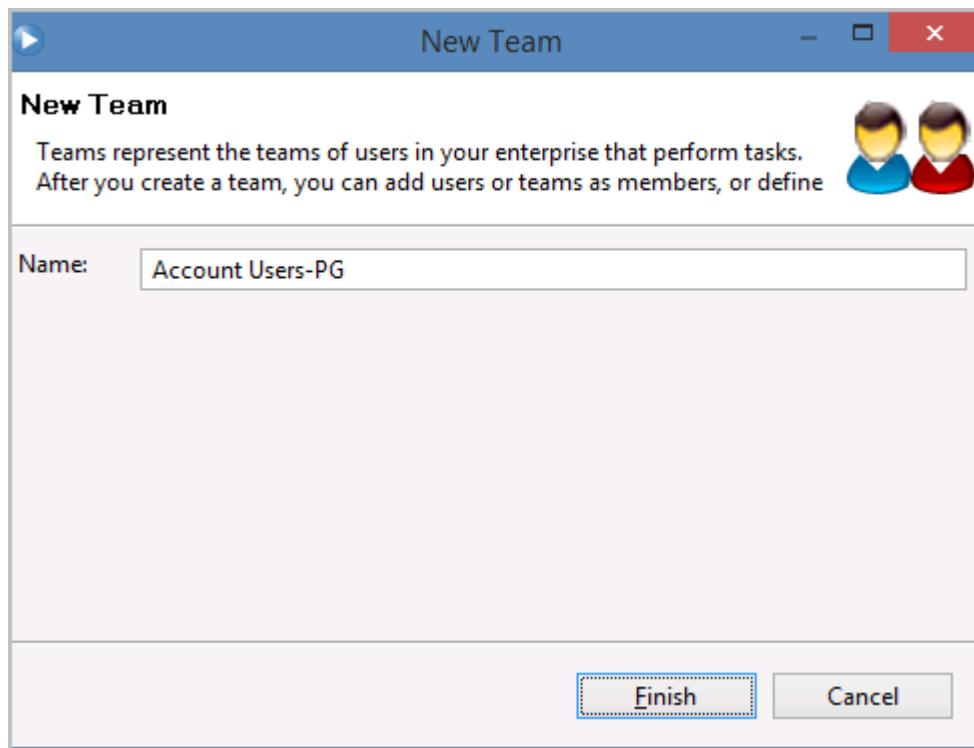
- \_\_\_ b. Click the **Participant** lane of the BPD to see the properties of the lane.



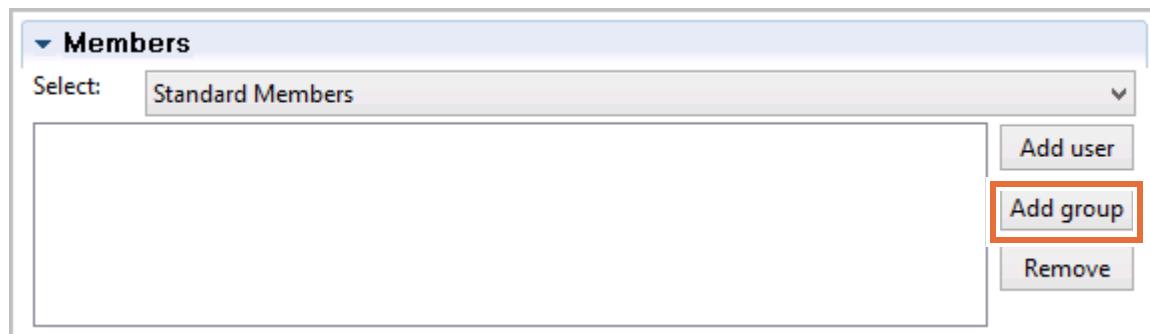
- \_\_\_ c. In the **Behavior** section of the **Properties** window, click **New** to create and assign the participant team to the lane.



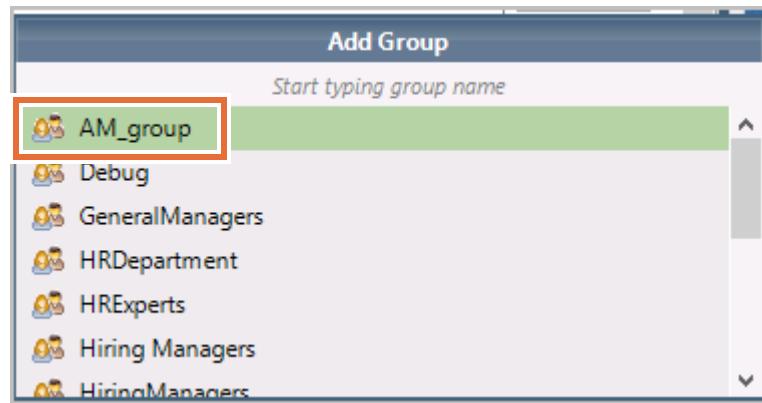
- \_\_\_ d. In the **New Team**, enter **Account Users-PG** as the Name and click **Finish**.



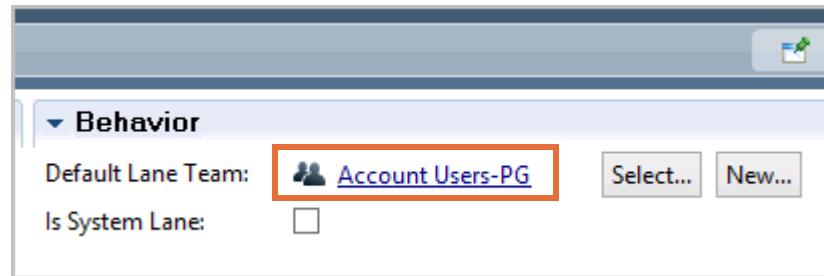
- \_\_\_ e. In the **Team** editor, click **Add group** in the **Members** section.



- \_\_ f. In the **Add Group** window, start typing **AM**, and it lists all groups with the initials AM. Click the **AM\_group**.



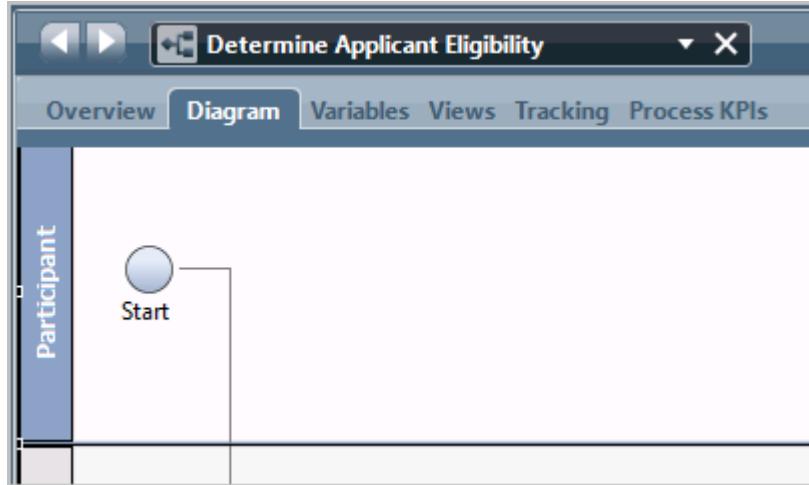
- \_\_ g. Press Ctrl+S to save your changes and Ctrl+W to close the editor.  
\_\_ h. Notice that the **Account Users-PG** team is added as the participant team in the **Behavior** section.



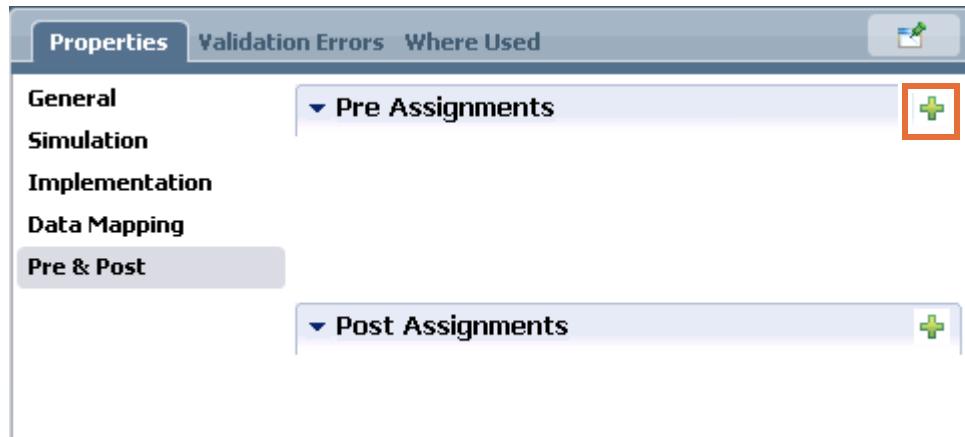
## Part 4: Test the business process diagram by using Playback and Inspector

In this part of the exercise, you test the business process diagram in the Inspector view by using Playback tools.

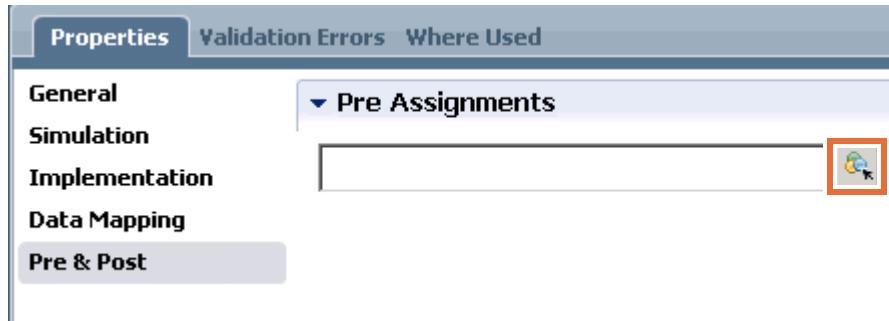
- 1. Before you can test the business process diagram, you must set some test data. For the purposes of this playback, test data is set in a post-assignment step of the start event.
  - a. Click the **Start** event in the business process diagram to select it.



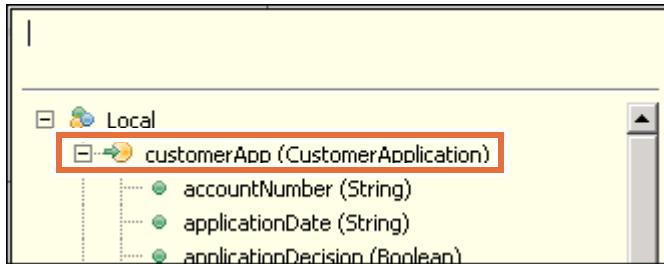
- b. In the **Properties** view, click the **Pre & Post** tab. With this selection, you can set pre-assignment and post-assignment options for an activity.
- c. In the **Pre Assignments** section, click the plus (+) sign to add a pre-assignment step.



- \_\_\_ d. On the left side of the equation, click the variable icon.



- \_\_\_ e. Double-click `customerApp` (`CustomerApplication`) to add it to the equation.



- \_\_\_ f. On the right side of the equation, type:

```
new tw.object.CustomerApplication()
```

This pre-assignment step initializes the input variable with a new `CustomerApplication` business object.

- \_\_\_ g. Click the plus sign (+) again to add a second pre-assignment command.  
 \_\_\_ h. On the left side of the equation, click the variable icon.  
 \_\_\_ i. Scroll down to find `customerAppOut` (`CustomerApplication`) and double-click `customerAppOut` (`CustomerApplication`) to add it to the equation.  
 \_\_\_ j. On the right side of the equation, type:

```
new tw.object.CustomerApplication()
```

This pre-assignment step initializes the output variable.

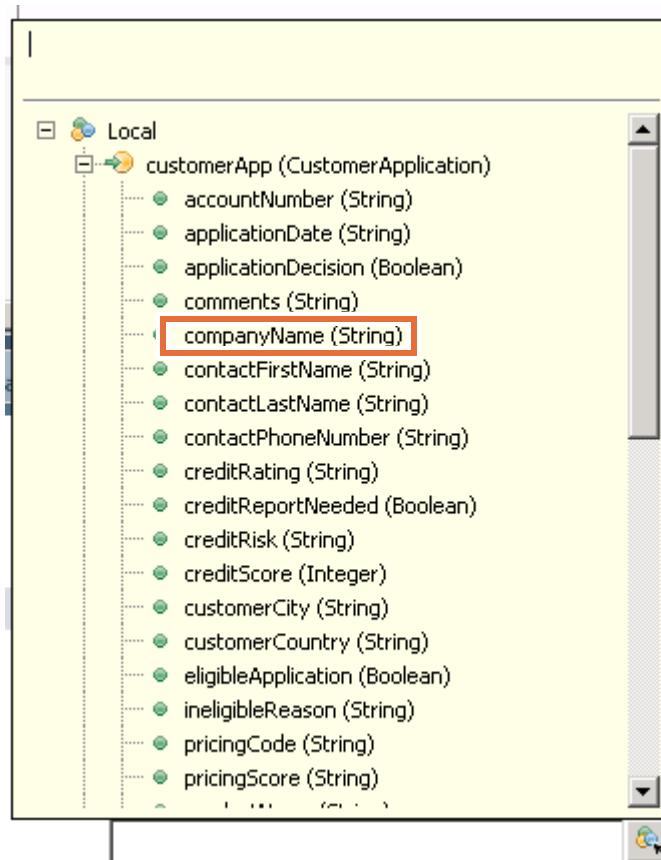


- \_\_\_ k. Save your work.

- \_\_ l. In the **Post Assignments** section, click the plus (+) sign to add a post-assignment step.

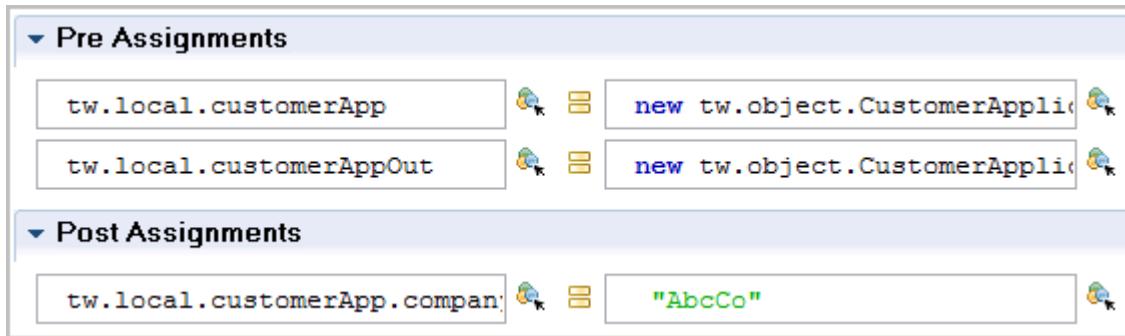


- \_\_ m. On the left side of the equation, click the variable icon.  
 \_\_ n. Double-click `companyName (String)` under `customerApp (CustomerApplication)` to add it to the equation.

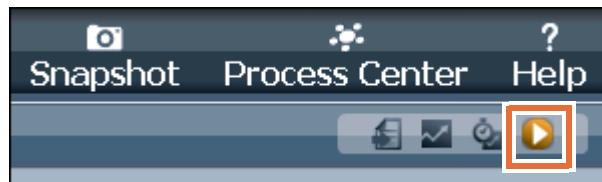


- \_\_ o. On the right side of the equation, type: "AbcCo"

You first test the process diagram by using the first path: the ineligible application.



- \_\_\_ 2. Save your work.
- \_\_\_ 3. In the upper-right corner of the process design view, click the **Run Process** icon to test the playback of this process diagram.



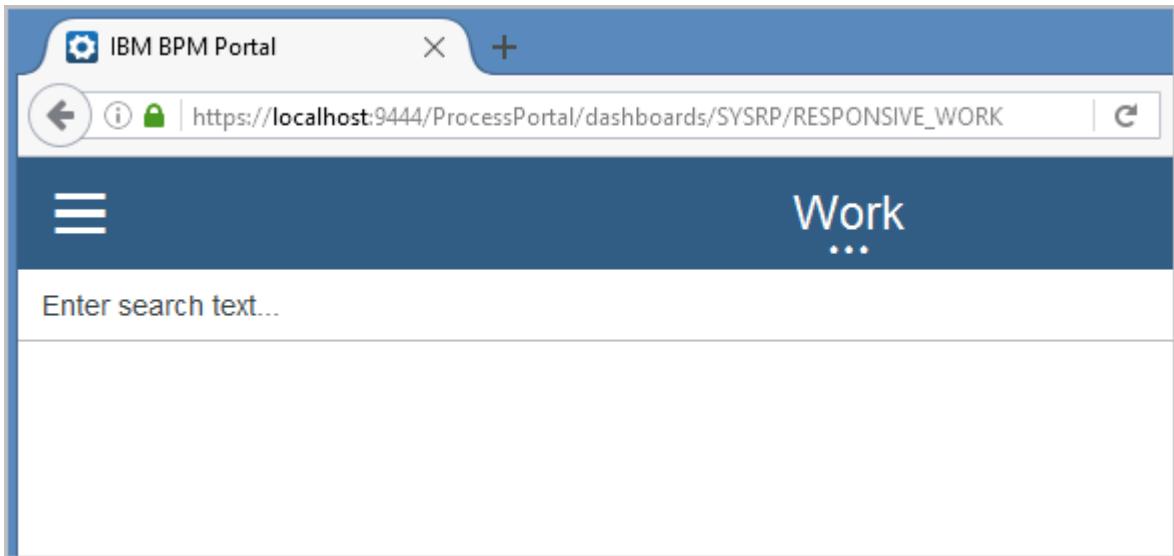
- \_\_\_ 4. In the **Switch View** window, click **Yes**.
- \_\_\_ 5. The process inspector stops at **View Customer information**.

The inspector moved from **Assess the Credit Record** to **Review the Customer Contract** activity, and from **Review the Customer Contract** activity to **View Customer Information**.

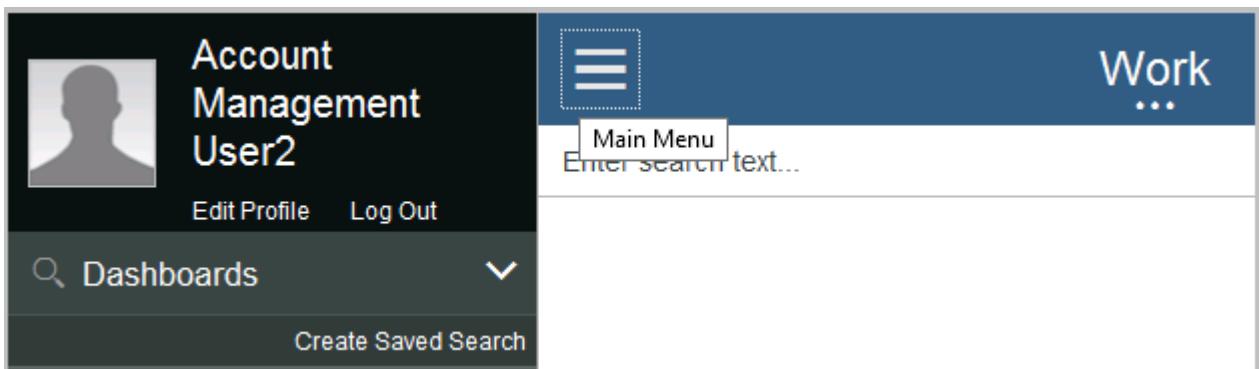
| Status   | Owner          | Subject                            | Priority | Due Date         | Task Id |
|----------|----------------|------------------------------------|----------|------------------|---------|
| Closed   | pcdeadmin      | Step: Assess the Credit Record     | Normal   | Sep 18, 2014 ... | 8       |
| Closed   | pcdeadmin      | Step: Review the Customer Contract | Normal   | Sep 18, 2014 ... | 9       |
| Received | (ROLE) Acco... | Step: View Customer Information    | Normal   | Sep 18, 2014 ... | 10      |

- \_\_\_ a. Open **IBM Process Portal** to work on the **View Customer Information** task. Start an instance of Firefox and enter <http://localhost:9081/portal> to open Process Portal.

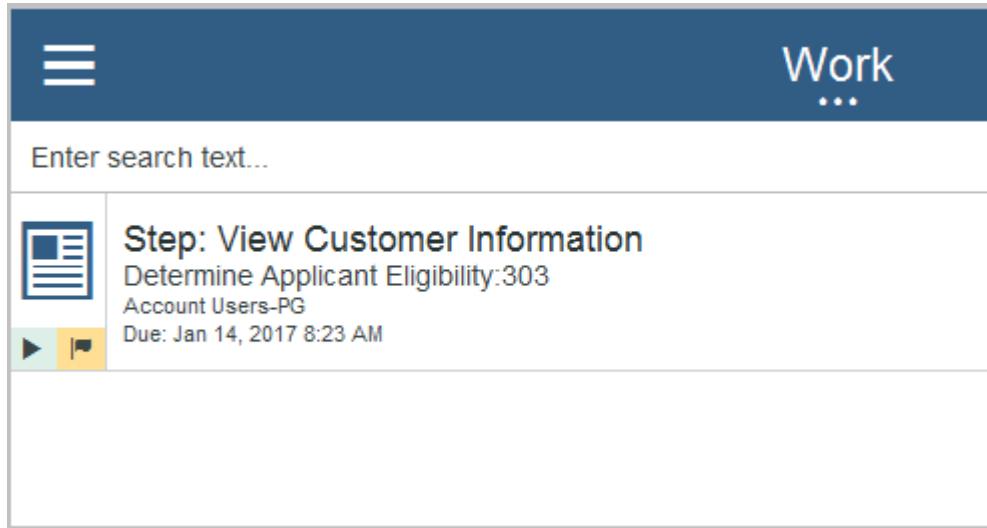
- \_\_ b. Log the user AM\_user2 in the Process Portal by entering `AM_user2` in the User ID field and `web1sphere` in the Password field. Notice that since `AM_user2` is not a member of the group that is assigned to this participant group, the **View Customer Information** task is not available for the `AM_user2` user.



- \_\_ c. Log out the user `AM_user2` from Process Portal. Click the drop-down arrow on the upper-right side and click **Logout**.



- \_\_\_ d. Log in to the Process Portal again now with the **AM\_user1** user. Enter `AM_user1` in the User ID field and `web1sphere` in the Password field. Since this user is a member of the group that is assigned to the participant group, now you can see the task that is ready to claim by this user. Click **Step: View Customer information**.



- \_\_\_ e. The Determine Applicant Eligibility page opens. Click the **Step: View Customer information** link at the bottom and then in the **Claim Task** window, click **Claim Task**.

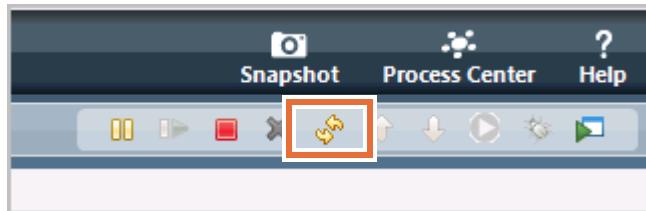
A screenshot of the "Determine Applicant Eligibility:303" page. The title is at the top. Below it is a "Data" section with a "▼ Data" button and the message "No details found.". Then there is a "Tasks" section with a "▼ Tasks" button. Underneath is a link: "Step: View Customer Information" with a gear icon. The entire page has a light gray background with horizontal sections separated by thin lines.

- \_\_\_ f. The **View Customer Information** coach opens. Click **OK** at the end of the page to complete the task.

## Step: View Customer Information ▾

|                         |                                         |
|-------------------------|-----------------------------------------|
| Credit Rating:          | <input type="text" value="F"/>          |
| Credit Report Needed:   | <input checked="" type="checkbox"/>     |
| Credit Risk:            | <input type="text"/>                    |
| Credit Score:           | <input type="text" value="0"/>          |
| Customer City:          | <input type="text" value="Madrid"/>     |
| Customer Country:       | <input type="text" value="Spain"/>      |
| Eligible Application:   | <input type="checkbox"/>                |
| Ineligible Reason:      | <input type="text" value="Bad credit"/> |
| Pricing Code:           | <input type="text" value="51"/>         |
| Pricing Score:          | <input type="text" value="31"/>         |
| Product Name:           | <input type="text" value="Tacks"/>      |
| Request Account Amount: | <input type="text" value="20,000"/>     |
| <b>Section Title</b>    |                                         |
| <b>OK</b>               |                                         |

- \_\_\_ g. Log out of the Process Portal and go back in the Inspector view of the Process Designer. Click the refresh icon from the Inspector toolbar.



- \_\_\_ h. Notice that all tasks that are related to the process are closed.

| Status | Owner     | Subject                            | Priority | Due Date         | Task Id |
|--------|-----------|------------------------------------|----------|------------------|---------|
| Closed | pcdeadmin | Step: Assess the Credit Record     | Normal   | Sep 18, 2014 ... | 8       |
| Closed | pcdeadmin | Step: Review the Customer Contract | Normal   | Sep 18, 2014 ... | 9       |
| Closed | AM_user1  | Step: View Customer Information    | Normal   | Sep 18, 2014 ... | 10      |

- \_\_\_ i. The test ends here.

- 6. *Do not* close the IBM Process Designer and *do not* stop IBM Process Center server. You are going to use them in the next exercise.

## End of exercise

---

# Exercise 13. Applying governance to process applications

## Estimated time

01:00

## Overview

In this exercise, you explore the sample governance process application, and enable governance to the process application to control its installation on the IBM Process Server. You also test the installation of a snapshot that goes through the governance process.

## Objectives

After completing this exercise, you should be able to:

- Create a governance process application
- Enable governance in a process application
- Test the governance of a snapshot installation

## Introduction

IBM Process Designer is a tool that is used for modeling and developing simple business processes in IBM Business Process Manager. In this exercise, you are provided with the sample governance process application. You explore this application and learn how to create your own governance process application according to the business requirement. You also see how the governance is enabled on a process application and test the entire scenario.

## Requirements

Completing the exercises for this course requires a lab environment. The environment includes the exercise support files, Mozilla Firefox, IBM Business Process Manager Advanced V8.5.7, IBM DB2 V10.1, RFHUTIL, WebSphere MQ V8.0, and the IBM Process Server V8.5.7 test environment.

## Exercise instructions

### Introduction to the governance process for the snapshot installation

Governance that is applied to the installation of a process application controls the installation of the process application snapshots on the process server. When governance is enabled on a process application, any request that is made to install a snapshot of the process application, from the IBM Process Center to IBM Process Server, undergoes certain approvals. These approvals are defined in the governance process. After all necessary approvals are in place, snapshots can get installed on the IBM Process Server; otherwise, they cannot.

The governance process that you create or reuse must have a dependency on the **System Governance toolkit (TWSYSG)**. The toolkit provides the resources that are required to build a governance process. The toolkit has various integration services to create a governance process such as Install Snapshot, Cancel Snapshot Installation, Get All Process Servers, Set Installation Status, and Get Installation Status.

In this exercise, you are provided with the sample governance process application **Basic Governance for Install Sample (BGIS)**.

You explore this sample governance process, which can help in understanding how this governance of snapshot installation works. You then enable governance in your process application and test the governance.

#### **Part 1: Explore the sample governance process application**

The sample governance process **Basic\_Governance\_for\_Install\_Sample - v1** is available for this exercise in the IBM Process Center repository. In this part of the exercise, you explore this sample governance process. In a real-world scenario, an administrator creates a process application that the governance uses and sets up a dependency on the System Governance toolkit. Only someone with administrative authority for the System Governance toolkit can create a dependency on that toolkit. The dependency of the process on System Governance toolkit must be in place before you set up the governance process application.

The sample process application is inside the centralized IBM Process Center repository. Therefore, to explore the sample process application in IBM Process Designer, you must first start the IBM Process Center cluster environment.

If you stopped the IBM Process Center environment in the previous exercise, start it again.

- 1. Start the Process Center deployment manager.
  - a. On your Windows desktop, select the shortcut that is titled: **Start Process Center deployment manager**. Double-click or press Enter to open the shortcut. It takes several minutes for the deployment manager to start. When the deployment manager starts, you are prompted to press any key to continue. Press any key to close the command window.

- \_\_\_ 2. Start the Process Center node agent.
  - \_\_\_ a. On your Windows desktop, select the shortcut that is titled: **Start Process Center node agent**. Double-click or press Enter to open the shortcut. It takes several minutes for node agent to start. When the node agent starts, you are prompted to press any key to continue. Press any key to close the command window.
- \_\_\_ 3. Start the Process Center Cluster.
  - \_\_\_ a. On your Windows desktop, select the shortcut that is titled: **Start Process Center Cluster**. Double-click or press Enter to open the shortcut. It takes several minutes (longer than the deployment manager and node agent processes) for the cluster to start. When the cluster starts, you are prompted to press any key to continue. Press any key to close the command window.
- \_\_\_ 4. If you stopped the IBM Process Designer in the previous exercise, start it again. On the Windows desktop, locate the icon that is labeled **Start IBM Process Designer**. Double-click the icon or press **Enter** to start IBM Process Designer.
  - \_\_\_ a. As the splash screen for IBM Process Designer loads, you are prompted to enter a user name and password to connect to the IBM Process Center. Enter `pcdeadmin` for **User Name** and `web1sphere` for **Password**.
  - \_\_\_ b. Click **Log In**. After a few moments, IBM Process Designer starts.
  - \_\_\_ c. If you are not in the Process Center perspective, click the **Process Center** link from the upper-right corner to go to the Process Center perspective.



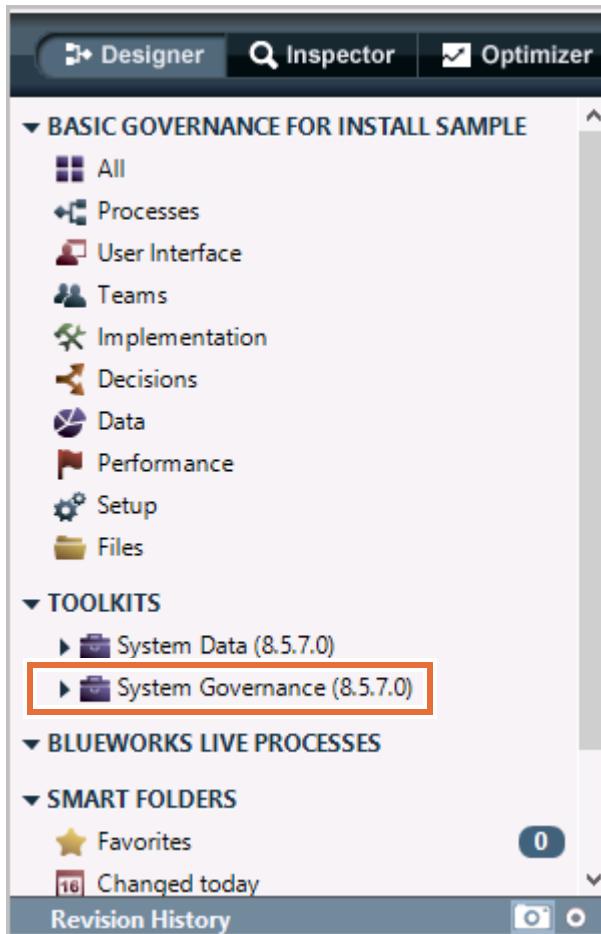
- \_\_\_ d. Make sure that you are in the **Process Apps** tab.
- \_\_\_ 5. Explore the **System Governance** toolkit in the sample governance process application.
  - \_\_\_ a. Locate the process app that is named `Basic Governance for Install Sample (BGIS)`.

- \_\_ b. Click the **Open in Designer** link that is associated with the process app. The process app opens in Designer view.

The screenshot shows the 'Process Apps' section of a software interface. At the top, there are tabs for 'Process Apps', 'Toolkits', 'Servers', and 'Admin'. Below the tabs, there is a search bar and a sorting dropdown set to 'Recently Updated'. There are two filter buttons: 'All' and 'Favorites'. The main area displays four process apps:

- Basic Governance for Install Sample (BGIS)** (highlighted with a red box around the entire row)
  - Last updated on 9/27/16 by pdeadmin
  - Open in Designer** (highlighted with a red box)
- Account Management (AM\_1012)**
  - Last updated on 9/27/16 by pdeadmin
  - Open in Designer**
- AIS Fault Handling (AISF)**
  - Last updated on 9/27/16 by pdeadmin
  - Open in Designer**
- Account Management (AM\_101)**
  - Last updated on 9/27/16 by pdeadmin
  - Open in Designer**

- c. The **System Governance (TWSYSG)** toolkit is listed under the **Toolkits**. This toolkit contains all the services that you need to create the business process definition of the governance process application. Since you are using already created sample governance process in the exercise, this dependency is already established for you.



### Note

Only the user with administrative authority can create the dependency of the governance process application on the **System Governance** toolkit. Before the business analyst or developer creates a business process definition (BPD) that implements a governance process, an administrator must create this toolkit dependency.

Since you are using an already created sample governance process in the exercise, this dependency is already established for you.

However, if this dependency is not created, to create the toolkit dependency in a new governance process application, you do the following steps:

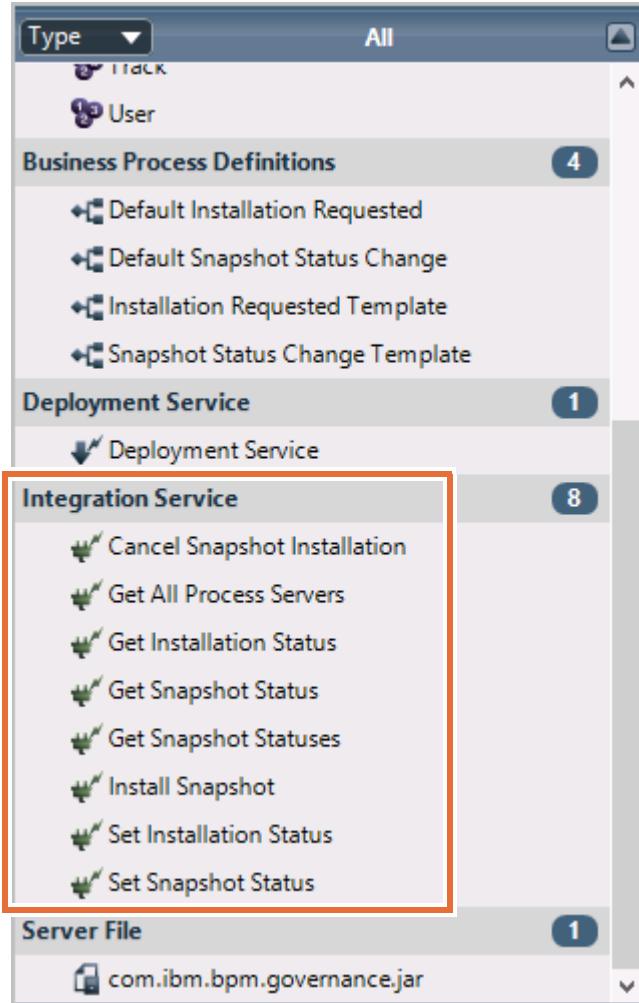
- Click the plus sign (+) to the right of **Toolkits**. The plus sign (+) is not visible by default; you hover over **Toolkits** to see the icon.

- Select the **8.5.7.0** under **System Governance**.



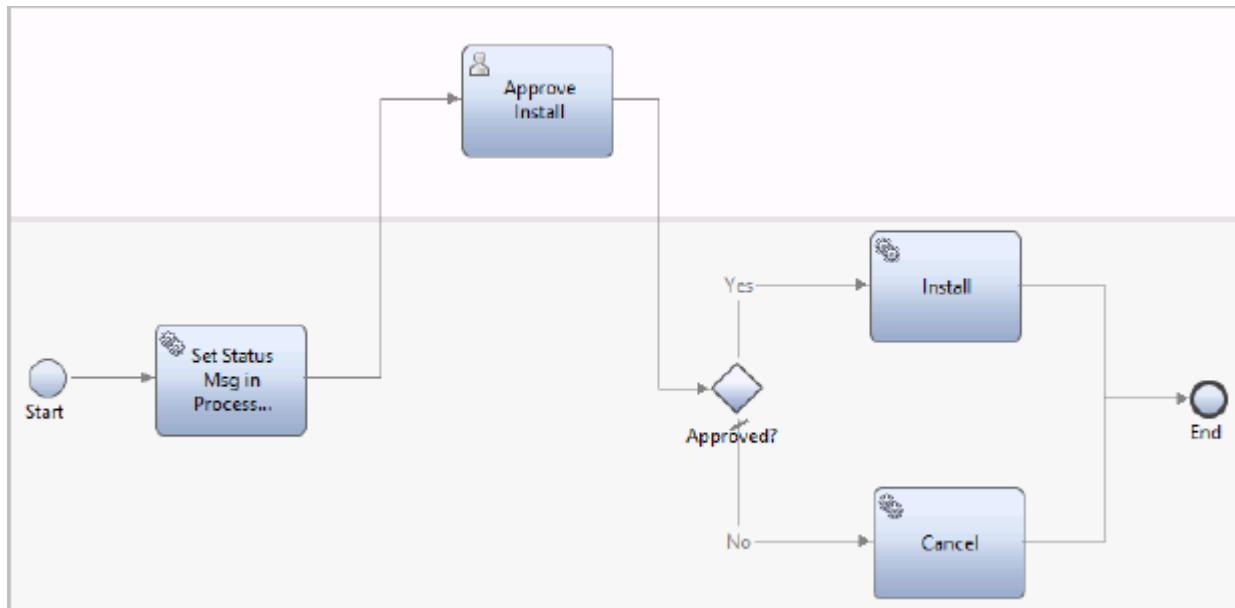
- 
- d. Expand the **System Governance(8.5.7.0)** toolkit.

- e. Select **All** from the list of available artifacts. Scroll to **Integration Service**. The integration services that are listed can be used to create a governance process BPD. For example, the **Install Snapshot** installs the snapshot to the specified process server, and **Cancel Snapshot Installation** cancels the installation of a snapshot that is already in progress.



\_\_ 6. Explore the governance process BPD.

- \_\_ a. In the Designer view of the **Basic Government For Install Sample** process app, click **Processes**. Double-click **Review Install Request** under **Business Process Definitions**. The BPD opens in the business process editor.



### Note

The BPD describes different activities that are involved in the installation approval process. You go through the implementation of each activity in rest of the lab material.

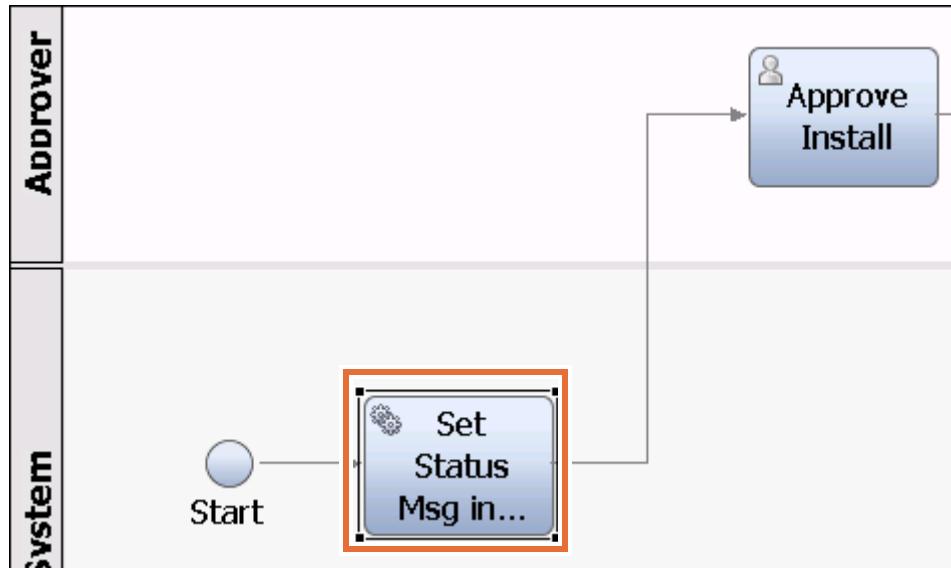
- \_\_ b. Select the **Variables** tab to see variables that are being used in the installation approval process.

| Variables   |  |
|-------------|--|
| Add Input   |  |
| Add Output  |  |
| Add Private |  |
| Link EPV    |  |
| Remove      |  |
| Move Up     |  |
| Move Down   |  |

**Note**

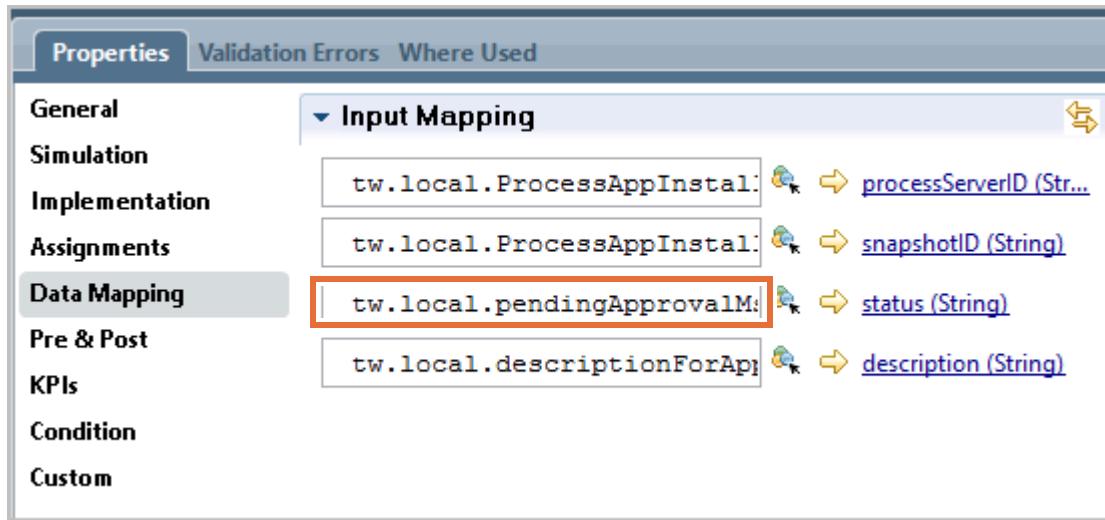
These variables contain all of the information that is related to process application snapshot installation. These values must be passed to the Review Install Request BPD. The approver might use this information to decide to approve a snapshot installation request. If time allows, click each variable to view the business object details.

- \_\_\_ c. Go back to the **Diagram** tab and select the first activity, **Set Status Msg in Process Center**, to explore its properties.

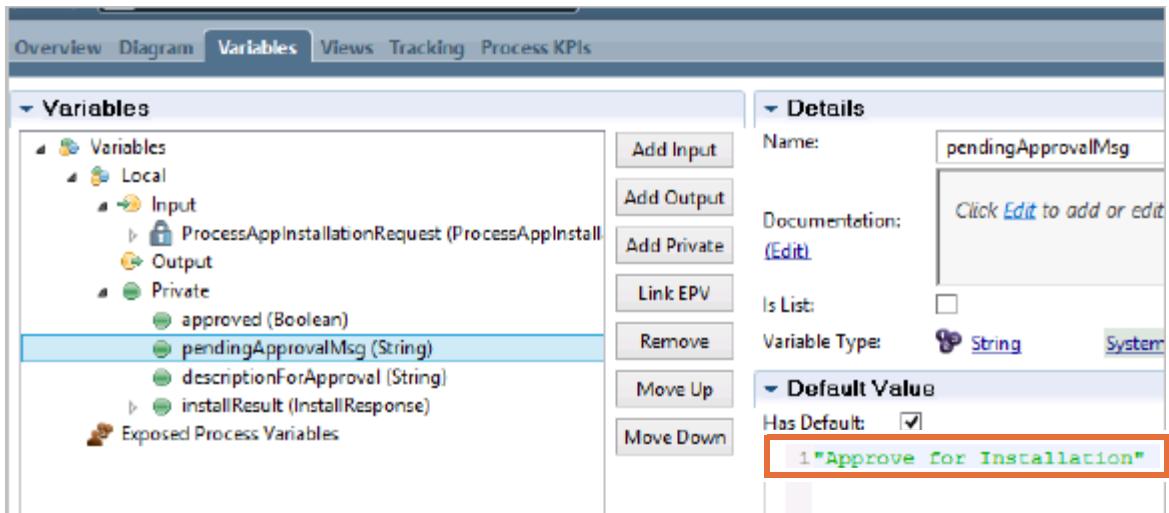


- \_\_\_ d. This service updates the installation status while an installation request is waiting for approval. When this service is called, the status updates immediately, without waiting for the status update or the installation to complete.

- e. At the bottom of the window, in the **Data Mapping** option in the Properties tab, observe the variables that are passed to the service. Here `tw.local.pendingApprovalMsg` contains the status message that flashes on the screen when the installation of the snapshot is requested. The variable is set to “**Approve for Installation**” in this sample.

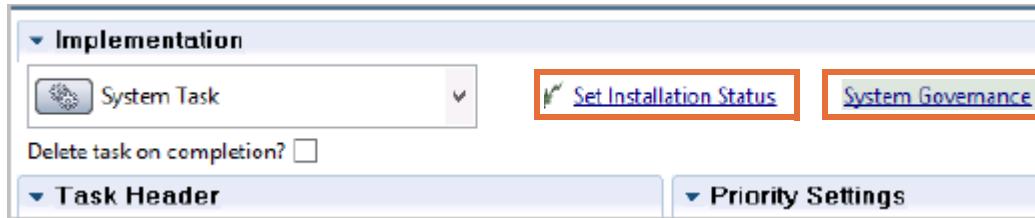


- f. To view the variable value, click the **Variables** tab and locate **pendingApprovalMsg** variable, and explore the value that is specified.

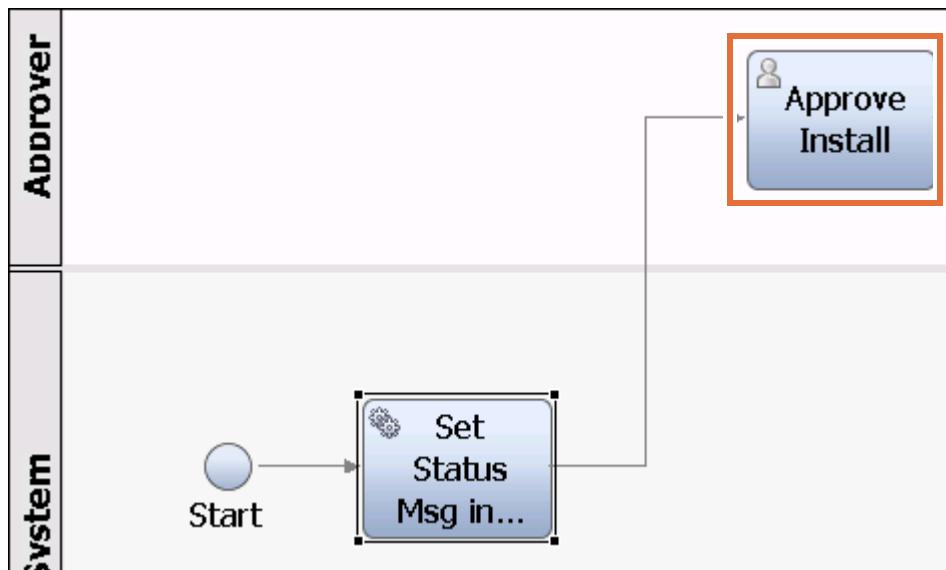


- g. Go back to the **Diagram** tab, select the first activity **Set Status Msg in Process Center**.

- h. In the **Properties** tab at the bottom, select the **Implementation** option. The **Set Installation Status** integration service inside the **System Governance** toolkit provides the implementation. The **Set Installation Status** integration service basically calls the Java API to set the installation status. If time allows, double-click the activity **Set Status Msg in Process Center** to explore more about its implementation details.



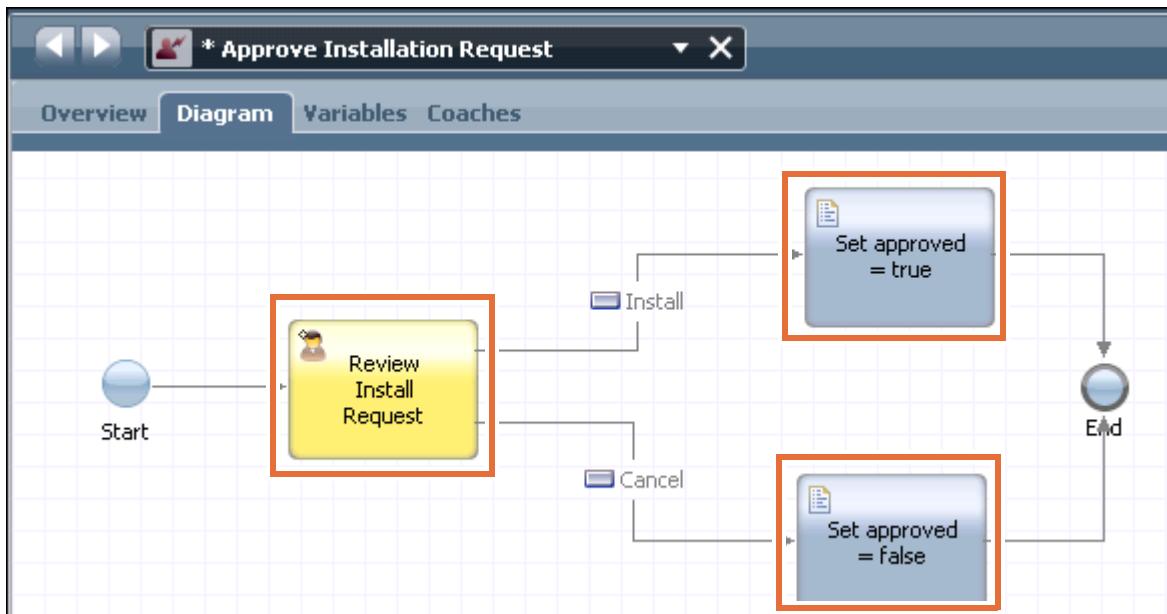
- i. Select the second activity **Approve Install**. When someone initiates an installation request for a snapshot of a process application that is under governance, the governance process passes a message with installation request details to a user task for installation approval. At the bottom of the window, select the **Properties** tab.



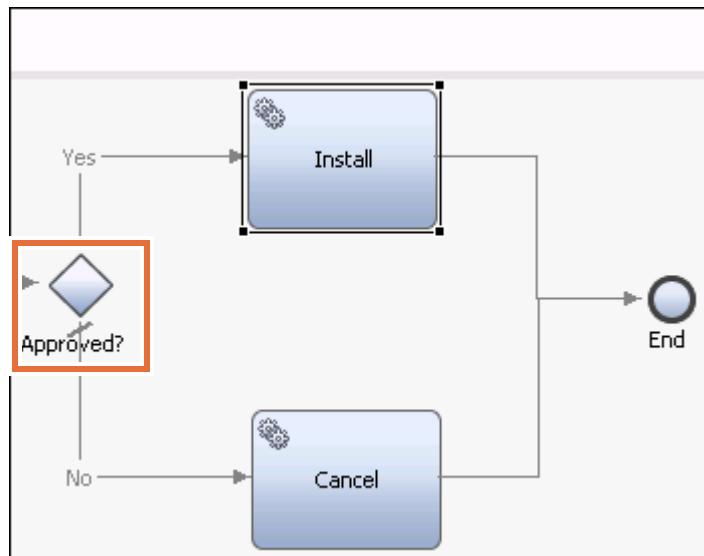
- j. In the **Implementation** option under Properties, notice that this activity is in the **Approver** lane so that a person or role can be assigned to this activity. The **Approve Installation Request** human service implements this activity.



- k. Double-click the **Approve Install** human service activity and explore its implementation. The **Approve Installation Request** human service contains a coach, **Review Install Request**, and a decision to install or cancel (**Set approved = true** or **Set approved = false**). So if the approver clicks **Install**, then **Set approved** is set to **true**; and if the approver clicks **Cancel**, then **Set approved** is set to **false**. In the real-world scenario, this implementation can be more complex according to the business requirement. For this exercise, the implementation is kept simple.



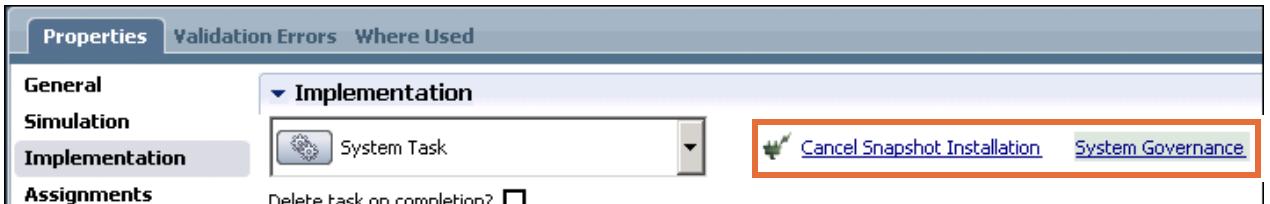
- l. Close the editor for the **Approve Installation Request** human service.
- m. The next item in the **Review Install Request** BPD is the **Approved?** decision gateway. If the approver approves the installation request that is explored in the previous step, the decision gateway directs the request to the **Install Snapshot** integration service, which the **System Governance** toolkit provides. If the approver cancels the request, the decision gateway directs the request to the **Cancel Snapshot Installation** integration service.



- \_\_\_ n. Select the **Install** activity and click the **Properties** tab at the bottom.
- \_\_\_ o. Select the **Implementation** option in Properties. Notice that the **Install Snapshot** service from the **System Governance** is selected.



- \_\_\_ p. Select the **Cancel** activity and click the **Properties** tab at the bottom.
- \_\_\_ q. Select the **Implementation** option in Properties. Notice that the **Cancel Snapshot Installation** service from the **System Governance** is selected.



## **Part 2: Enabling governance for a process application**

When the governance process that you explored in the previous part is in place, you can enable governance on your process application. When governance is enabled, a snapshot of the process application can be installed on a server only when all the approvals that are registered in the governance process are provided.

Only the person with administrative authority can enable governance in a process application.

- \_\_\_ 1. Enable governance on a **Hiring Sample** process application. Click the **Process Center** icon in the upper right to return to the perspective.



- \_\_\_ 2. Click the **Process Apps** tab.

- 3. Locate the process application that is named **Basic Governance for Install Sample (BGIS)** and click it.

The screenshot shows the 'Process Apps' interface with a navigation bar at the top labeled 'Process Apps', 'Toolkits', 'Servers', and 'Admin'. Below the bar, a list of applications is displayed in a table-like format. The first row, which contains the application name 'Basic Governance for Install Sample (BGIS)' along with its icon, a star rating, and a question mark icon, is highlighted with a red rectangular box. The other three rows in the list show 'Account Management (AM\_1012)', 'AIS Fault Handling (AISF)', and another 'Account Management (AM\_101)' entry, each with their respective icons, star ratings, and question marks.

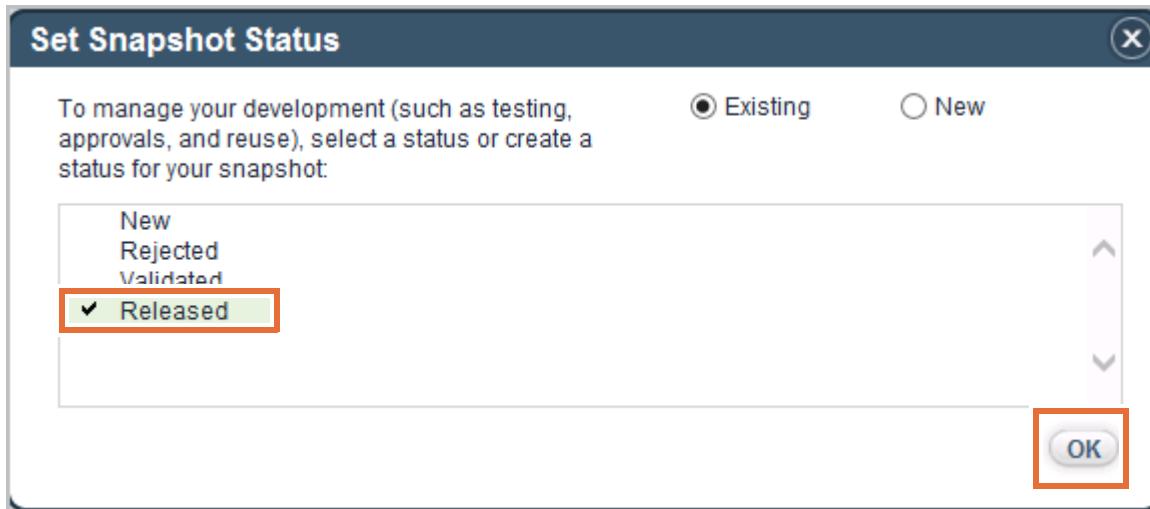
## Information

You do **not** have to click the **Open in Designer** link in front of the application.

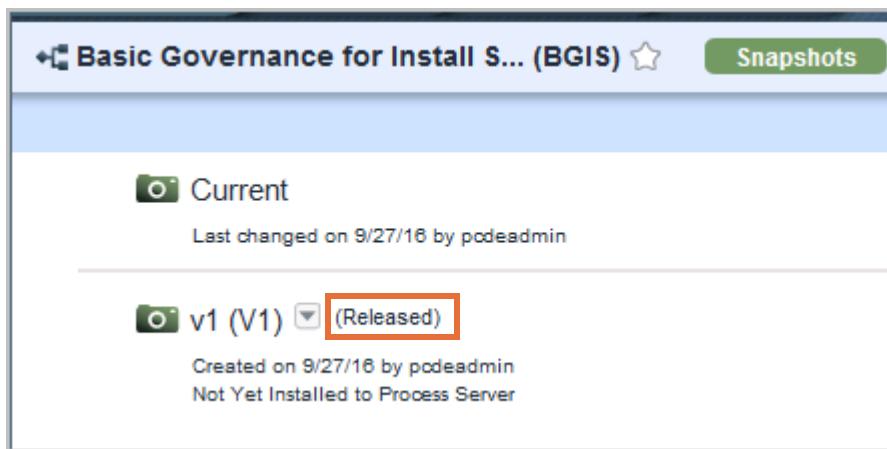
- 4. In the **Snapshots** tab, click the down arrow and select **Status** to change the status.

The screenshot shows the details page for the 'Basic Governance for Install S... (BGIS)' application. At the top, there is a header with the application name and a star rating, followed by a green button labeled 'Solutions'. Below the header, there is a section titled 'Current' with a 'Last changed on 9/27/16 by pdeadmin' message. Underneath this, there is a list of snapshots, starting with 'v1 (V1)'. To the right of 'v1 (V1)', there is a small dropdown arrow icon, which is also highlighted with a red box. A context menu is open over this dropdown, with the 'Status' option highlighted by a red box. Other options in the menu include 'Edit', 'Clone', 'Activate', 'Archive', and 'Generate Migration Policy'. The rest of the page is mostly blank space.

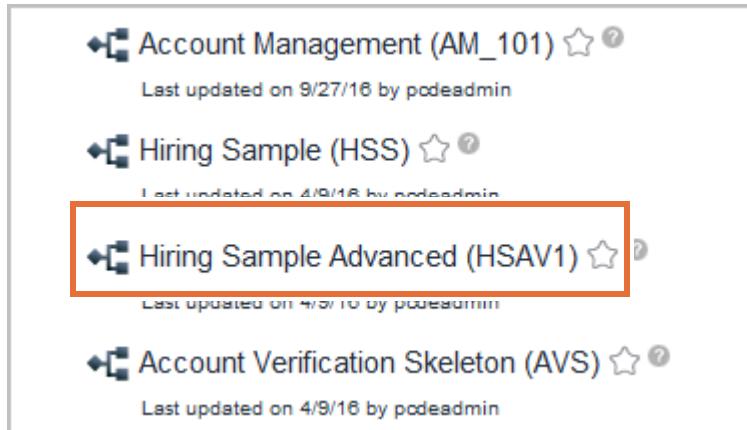
- 5. In the **Set Snapshot Status** screen, select **Released** and click **OK**.



- 6. Now, the governance process application should show the snapshot status as **Released**.



- 7. Click the **Process Apps** tab.  
— 8. Locate the process application named **Hiring Sample Advanced (HSAV1)** and click it.



- 9. Click the **Governance** tab and click **Change** in front of **Installation Requested (Default)**.

The screenshot shows the 'Governance' tab selected in the top navigation bar. Below it, two items are listed: 'Installation Requested (Default)' and 'Snapshot Status Change (Default)'. Each item has a 'Change' button to its right. The 'Change' button for 'Installation Requested (Default)' is highlighted with a red box.

- 10. In the **Change Installation Requested** screen, select **Review Install Request** and click **OK**. When that option is selected, any attempt to install a snapshot of that process application to a server starts a governance process to gather approvals and then install the snapshot.

The screenshot shows the 'Change Installation Requested' dialog box. It displays a list of governance options. The 'Review Install Request' option is highlighted with a red box. The text next to it indicates it is a 'Version v1 (Snapshot) created on 1/14/17 and found in Basic Governance for Install Sample'.

### **Part 3: Testing the governance that is applied to the process application**

In this part of the exercise, you test the governance that is applied to the installation of the **Hiring Sample Advanced (HSAV1)** snapshot. To test this feature, both the servers **IBM Process Center** and **IBM Process Server** must be running. You test the installation of the snapshot from the IBM Process Center to the IBM Process Server.

- 1. If IBM Process Server is stopped in the previous exercise, you must start it again.

**Note**

This test requires that both the IBM Process Center and IBM Process Server server environments must run concurrently. Running two WebSphere Application Server profiles concurrently places a tremendous strain on system resources, particularly in a virtual environment. Expect a severe impact on performance when you run this test.

- a. If the IBM Process Server is running from the previous exercise, then go to the next step. Otherwise, on your Windows desktop, select the shortcut that is titled: **Start UTE Process Server**. Double-click or press **Enter** to start the shortcut. It takes several minutes for the process to start. When the process server starts, the command window closes.
- 2. In the IBM Process Center console, click the **Snapshots** tab of the **Hiring Sample Advanced (HSAV1)**.

The screenshot shows the IBM Process Center interface. At the top, there's a header bar with the title "Hiring Sample Advanced (HSAV1)" and three tabs: "Solutions" (highlighted in green), "Snapshots" (highlighted in red), and "History". Below the tabs, there's a search bar labeled "Sort Snapshots By:" with dropdown menus for "Date" and "All".  
  
The main content area displays two snapshot entries:

- Current**: Last changed on 4/9/16 by pdeadmin. Status: Not Yet Deployed to Process Center Server.
- AHS goes BLUE v2 (AHSGBLU)**: Created on 4/9/16 by pdeadmin. Status: 5 Validation Errors, Not Yet Deployed to Process Center Server, Not Yet Installed to Process Server. This entry is marked as "(New)".

- 3. Click the **Install** link in front of the **Standard Hiring Sample** snapshot. Notice that since this snapshot is not yet deployed to the IBM Process Server, it is marked with the text **Not Yet Installed to Process Server**.

**Hiring Sample Advanced (HSAV1)**

Snapshots History Manage Governance

Sort Snapshots By: Date All | Installed | Deployed | Archived

Current

Last changed on 4/9/16 by pcdeadmin  
Not Yet Deployed to Process Center Server

AHS goes BLUE v2 (AHSGBLU) (New)

Created on 4/9/16 by pcdeadmin  
5 Validation Errors  
Not Yet Installed to Process Server

Export Install

- 4. In the **Install Snapshot to Server** window, select a server where the snapshot must be installed. Click the **UTEServer** link as the online server.

Install Snapshot to Server

Select a server to Install snapshot AHS goes BLUE v2 to:

UTEServer (ws2012r2x64)  
TEST - Status: Connected

- 5. When the server is selected, a green check mark is displayed in front of the **UTEServer** server. Click **Install**.
- 6. Notice that the status message is displayed just below the snapshot to provide the status of the installation progress.

**Hiring Sample Advanced (HSAV1)**

Snapshots History Manage Governance

Sort Snapshots By: Date All | Installed | Deployed | Archived

Current

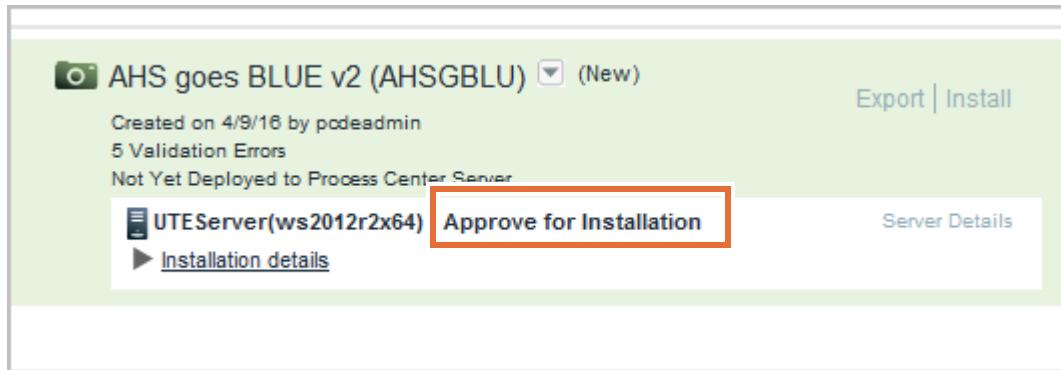
Last changed on 4/9/16 by pcdeadmin  
Not Yet Deployed to Process Center Server

AHS goes BLUE v2 (AHSGBLU) (New)

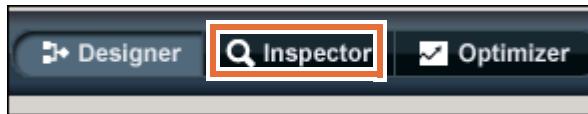
Created on 4/9/16 by pcdeadmin  
5 Validation Errors  
UTEServer(ws2012r2x64) - Installation in progress

Export | Install

- 7. Wait for few seconds and observe that the status message is updated as **Approve for Installation**. Recall that this message is same that you explored earlier in the **Set Installation Status** integration service in the **Review Install Request BPD** in this exercise.



- 8. Click the **Process Apps** tab.
- 9. Locate the process app that is named **Basic Governance for Install Sample (BGIS)**.
- 10. Click the **Open in Designer** link that is associated with the process app. The process app opens in Designer view.
- 11. Click the **Inspector** tab to open the Inspector view.



- 12. Click the refresh icon from the Inspector toolbar.



- 13. Click the **Review Install Request** instance from the upper-left pane.

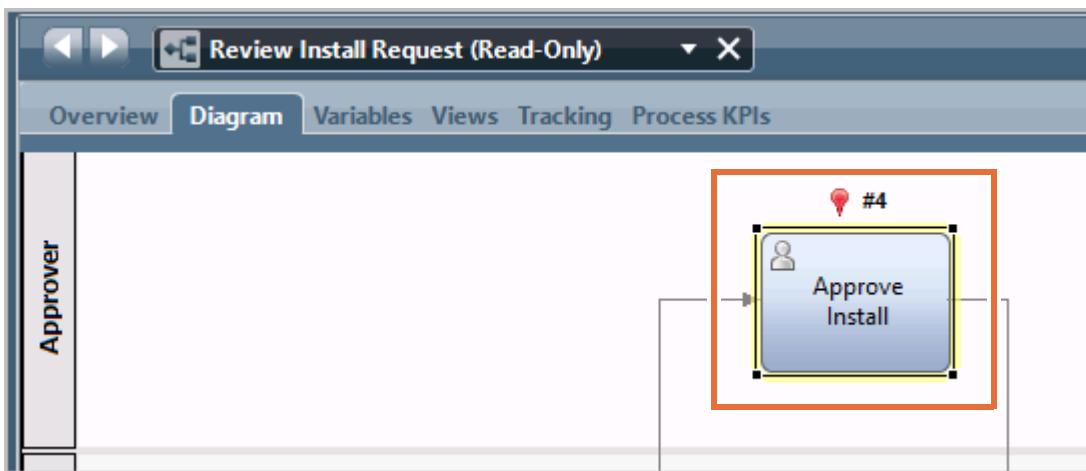
| Process Instances             |          | Services in Debug      |        |  |
|-------------------------------|----------|------------------------|--------|--|
| Instance Name:                | Status:  | All                    |        |  |
| Instance Name                 | Snapshot | Process                | Status |  |
| Basic snapshot governance:305 | v1       | Review Install Request | Active |  |
|                               |          |                        |        |  |

- 14. Again, click the refresh icon from the Inspector toolbar.

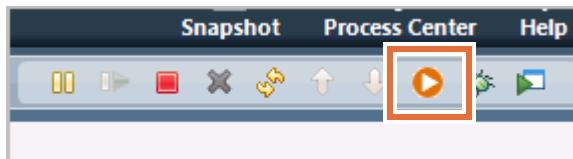
- 15. Examine the upper-right pane; it shows that the **Set Status Msg in Process Center** is successfully run and closed. Now the **Approve Install** activity is ready to be run.

| Status   | Owner       | Subject                              | Priority |
|----------|-------------|--------------------------------------|----------|
| Closed   | pcdead...   | Step: Set Status Msg in Process C... | Normal   |
| Received | (ROLE) A... | Step: Approve Install                | Normal   |

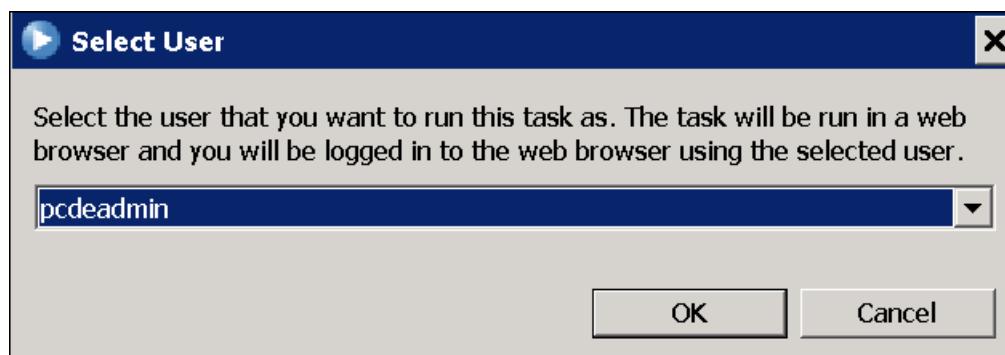
- 16. Examine the lower-right diagram pane; it shows that the business process definition is running and the token is waiting at the **Approve Install** activity.



- 17. Select the **Approve Install** activity in the upper-right pane and click **Runs the selected task**.



- 18. In the **Select User** window, select **pcdeadmin** and click **OK**.



- 19. If an insecure connection message is displayed, then add the exception. If prompted to enter credentials, use **pcdeadmin** and **web1sphere** to log in.

- 20. Wait for few seconds until the browser window displays process application information. Click **Step:Approve Install** to approve the installation of the snapshot on IBM Process Server. Click Claim task. If the task is already claimed, then skip to the next step.
- 21. Click **Install** to install the snapshot on IBM Process Server.

**Step: Approve Install**

**Approve Installation of Process Application to Process Server**

| <b>Snapshot</b>                                  |                                                |
|--------------------------------------------------|------------------------------------------------|
| Process Application Name: Hiring Sample Advanced |                                                |
| Snapshot Name: AHS goes BLUE v2                  |                                                |
| <b>Already Installed Snapshots</b>               | <b>Migration Action for existing instances</b> |
| <b>Initiated by</b>                              |                                                |
| Name: pcdeadmin<br>Date Initiated: Jan 14, 2017  |                                                |
| <b>Server</b>                                    |                                                |
| Name: UTEServer                                  | Online: true                                   |
| type: Test                                       | Is Available: true                             |
| <b>Approval to Install</b>                       |                                                |
| <b>Install</b> <b>Cancel</b>                     |                                                |

- 22. Close the web page when the service finishes.
- 23. You are back in the Inspector view of the Process Designer. Wait for a minute or so; since both servers are running, the response can be slow. Click the refresh icon from the Inspector toolbar. You might have to click the refresh icon few times to see the status change. Notice that the **status** of all the tasks that are related to the process is **closed**.

| Status | Owner     | Subject                              | Priority |
|--------|-----------|--------------------------------------|----------|
| Closed | pcdead... | Step: Set Status Msg in Process C... | Normal   |
| Closed | pcdead... | Step: Approve Install                | Normal   |
| Closed | pcdead... | Step: Install                        | Normal   |

- 24. Click the **Process Center** icon in the upper right to return to the perspective.
- 25. Click the **Hiring Sample Advanced (HSAV1)** process application.

- 26. In the **Snapshots** tab, notice that now the snapshot **Hiring Sample Advanced** is installed on the UTEServer (Process Server) and status is changed to **Currently Installed**.

The screenshot shows the IBM Process Center application window. At the top, there are tabs: Process Apps, Toolkits, Servers, Admin, Snapshots (which is highlighted in green), and History. Below the tabs, the title bar says '+ Hiring Sample Advanced (HSBV1) ☆'. Underneath, there's a dropdown menu set to 'Current'. It shows two entries:

- Hiring Sample Advanced (HSBV1)**: Last changed on 4/9/16 by pdeadmin. Not Yet Deployed to Process Center Server.
- AHS goes BLUE v2 (AHSGBLU)**: (New). Created on 4/9/16 by pdeadmin. 5 Validation Errors. Not Yet Deployed to Process Center Server. Currently Installed: **UTEServer(ws2012r2x64) - 0 instances**. There is a link to [Installation details](#).

- 27. Do not shut down IBM Process Center Server as you need it in the next exercise.  
 — 28. Close all open windows.  
 — 29. (Optional) Stop the **IBM Process Server** server.
- a. On the desktop, double-click the shortcut titled **Stop UTE Process Server**. It takes several minutes for process server to stop. When the process stops, you are prompted to press any key to continue. Press any key to close the command window.

## End of exercise

---

# Exercise 14. Integrating other applications with IBM Integration Designer

## Estimated time

01:00

## Overview

In this exercise, you explore the procedures for integrating IBM Business Process Manager assets with other applications, such as IBM Business Monitor and IBM Case Management.

## Objectives

After completing this exercise, you should be able to:

- Create a monitor model
- Import and explore a monitor model
- Associate a monitor model with a process application

## Requirements

Completing the exercises for this course requires a lab environment that includes the exercise support files, Mozilla Firefox, IBM Integration Designer Advanced V8.5.7, IBM DB2 V10.1, and the IBM Process Server V8.5.7 test environment.

## Exercise instructions

In this exercise, you do the steps that are necessary to integrate IBM Business Process Manager with other IBM products, such as IBM Business Monitor. Because this product is not installed in the lab environment, an effective test is not possible. However, the steps that you do in this exercise demonstrate the steps that are required and the appropriate practices for implementing IBM Business Process Manager in a greater solution.

# Integrating IBM Business Process Manager with IBM Business Monitor

The IBM Business Monitor development toolkit provides the tools for creating monitor models that can be transformed into executable code for IBM Business Monitor. The toolkit runs on IBM Integration Designer and includes the Monitor model editor and the Business Monitor test environment.

Business Monitor models are XML documents that specify how information can be extracted from events at runtime and collected, combined, and stored for representation on a dashboard. The completed monitor model contains a monitor details model, key performance indicator (KPI) model, dimensional model, visual model, and event model. Starting to create a monitor model can be done in three ways:

- Create a monitor model from scratch in the Monitor model editor
- Import a preliminary monitor model from another IBM tool. This model provides high-level descriptions of KPIs and business-relevant metrics and process diagrams that can be viewed in the IBM Business Monitor dashboards.
- Generate a monitor model from an IBM Process Server or WebSphere Enterprise Service Bus application. This model provides low-level information such as event definitions, inbound events, and correlation expressions, which are based on the events or common monitoring templates that you choose. You refine and extend the monitor model in the Monitor model editor. You must be working in IBM Integration Designer to use this option.



## Important

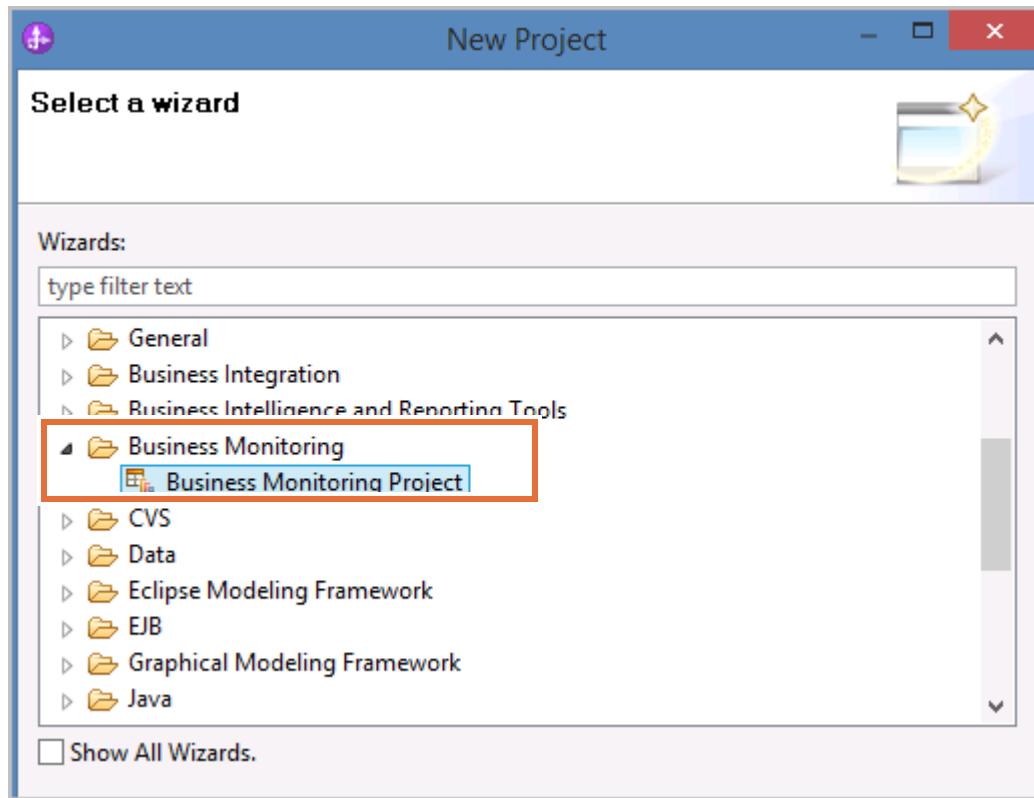
This exercise is not intended to teach you how to build a complete monitor model with key performance indicators. It is intended to demonstrate how tools in IBM Integration Designer can be used to generate monitor models. It is also intended to demonstrate good practices when you integrate IBM Integration Designer with IBM Business Monitor.

## **Part 1: Create a monitor model**

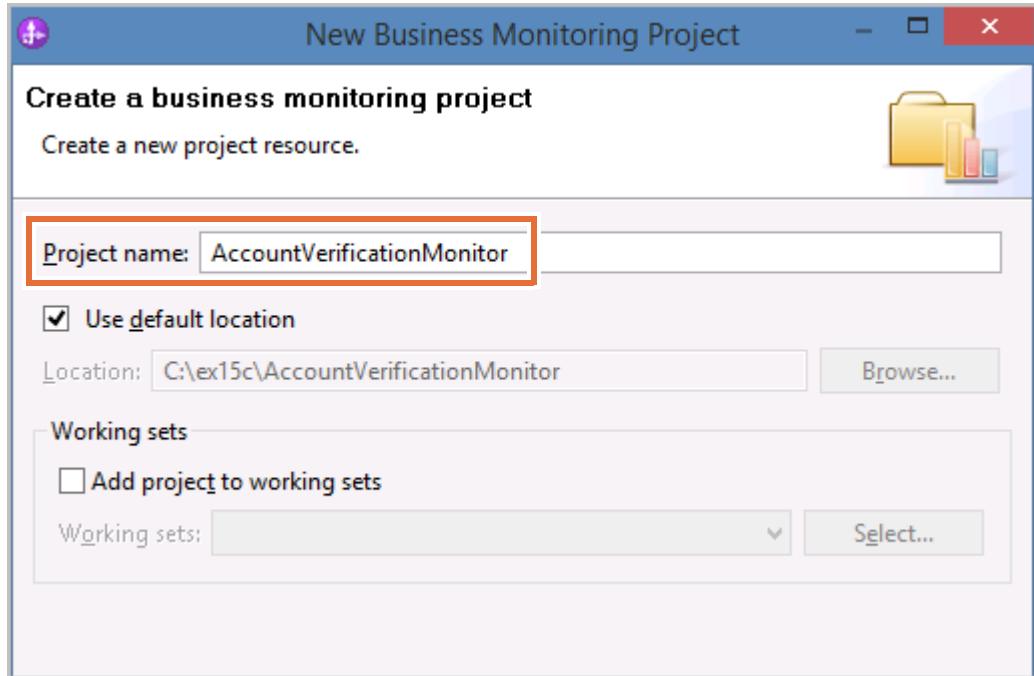
In this section of the exercise, you create a monitor project and a model.

- 1. Open the Exercise 14 workspace.
  - a. On your desktop, open the folder that is labeled **Exercise Shortcuts**.
  - b. Double-click the shortcut that is labeled **Exercise 14**.
  - c. Close the **Getting Started** tab.
- 2. Switch to the Business Monitoring perspective. Click **Window > Open Perspective > Business Monitoring** or **Window > Open Perspective > Other...> Business Monitoring**.
- 3. Create a monitoring project that monitors the `AccountVerification` business process artifacts.
  - a. Click **File > New > Project**.

- \_\_ b. In the **New Project** wizard, expand **Business Monitoring** and select **Business Monitoring Project**.

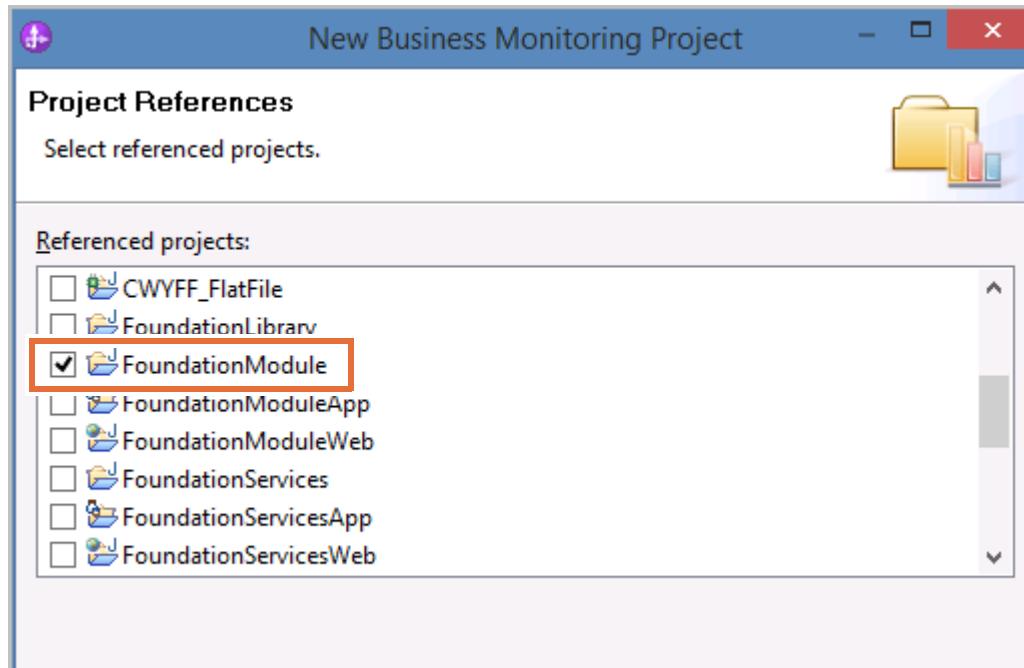


- \_\_ c. Click **Next**.  
\_\_ d. Set the project name to: AccountVerificationMonitor



- \_\_ e. Click **Next**.

- \_\_ f. In the **Project References** window, select the **FoundationModule** project.



- \_\_ g. Click **Finish**.  
\_\_ 4. Close the **Technology Quickstart** for monitor models.

**Help for developing monitor models**

### Featured help topics

To learn the tasks that you perform while developing monitor models for monitoring business processes, follow these links.

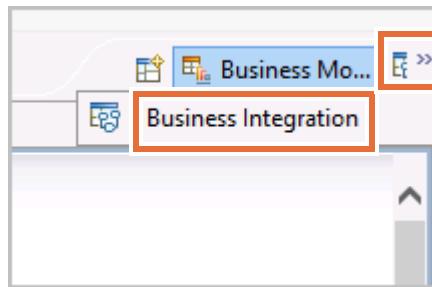
- ❑ [Step 1. Create a monitor model](#)  
Create a new monitor model, import a model, or generate a model from an application.
- ❑ [Step 2. Test the model](#)  
After creating the monitor model, test it and check the results in Business Space.
- ❑ [Step 3. Debug the model](#)  
Send an event to the debugger and follow each processing step as the model runs.



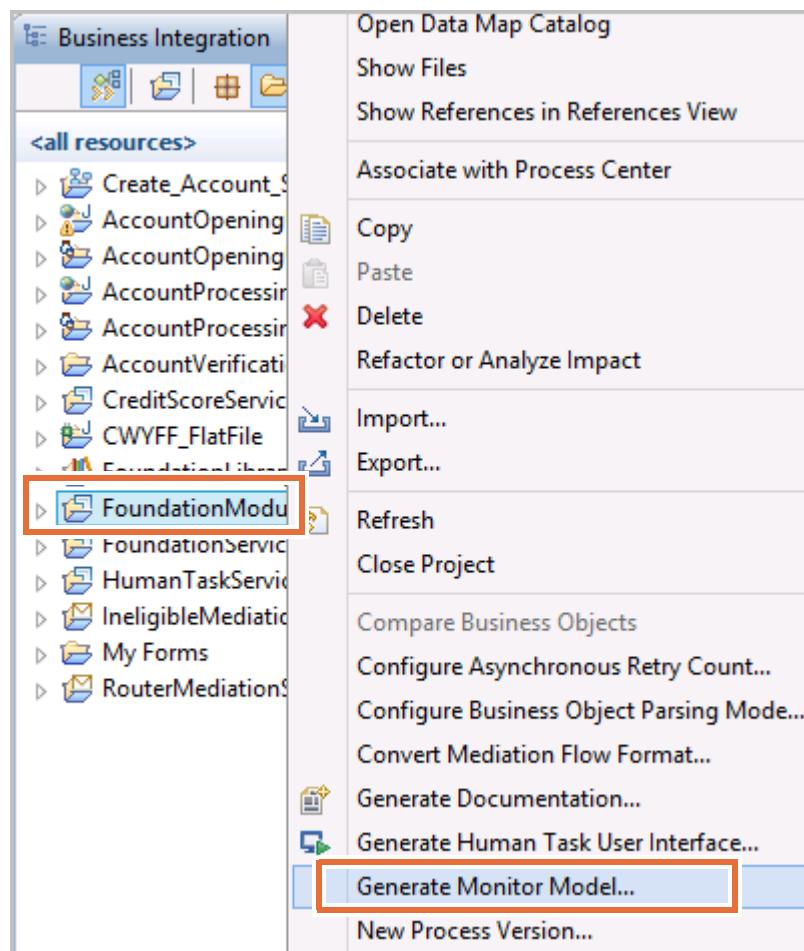
### Information

The Technology Quickstart is a useful tool that provides documentation and help for creating and working with monitor models. The tool provides help in a useful step-by-step approach.

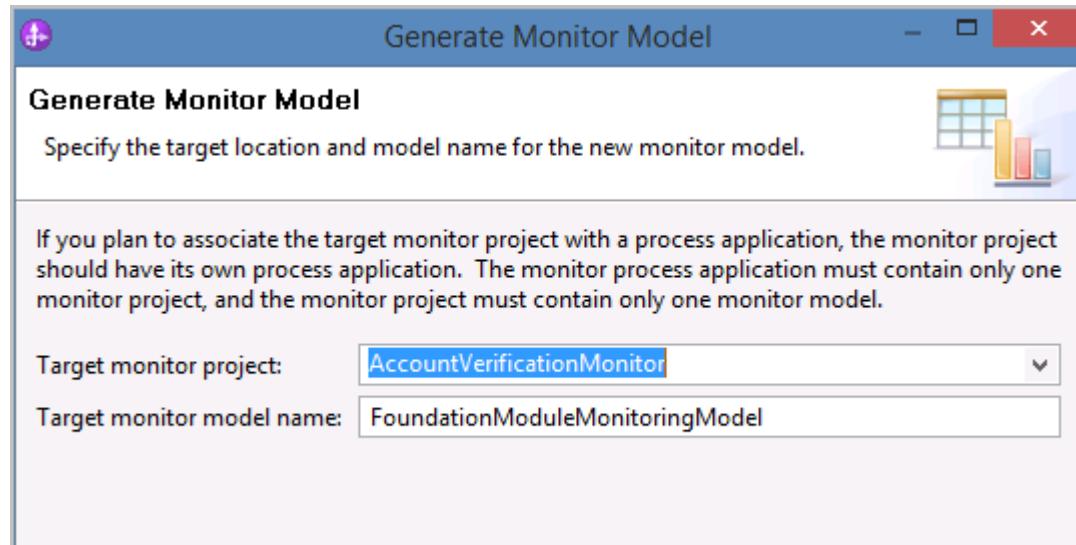
- 5. Generate a monitor model from IBM Integration Designer assets.
- a. Return to the Business Integration perspective. You can either click the Business Integration icon in the upper right of IBM Integration Designer or click **Window > Open Perspective > Other > Business Integration (default)**.



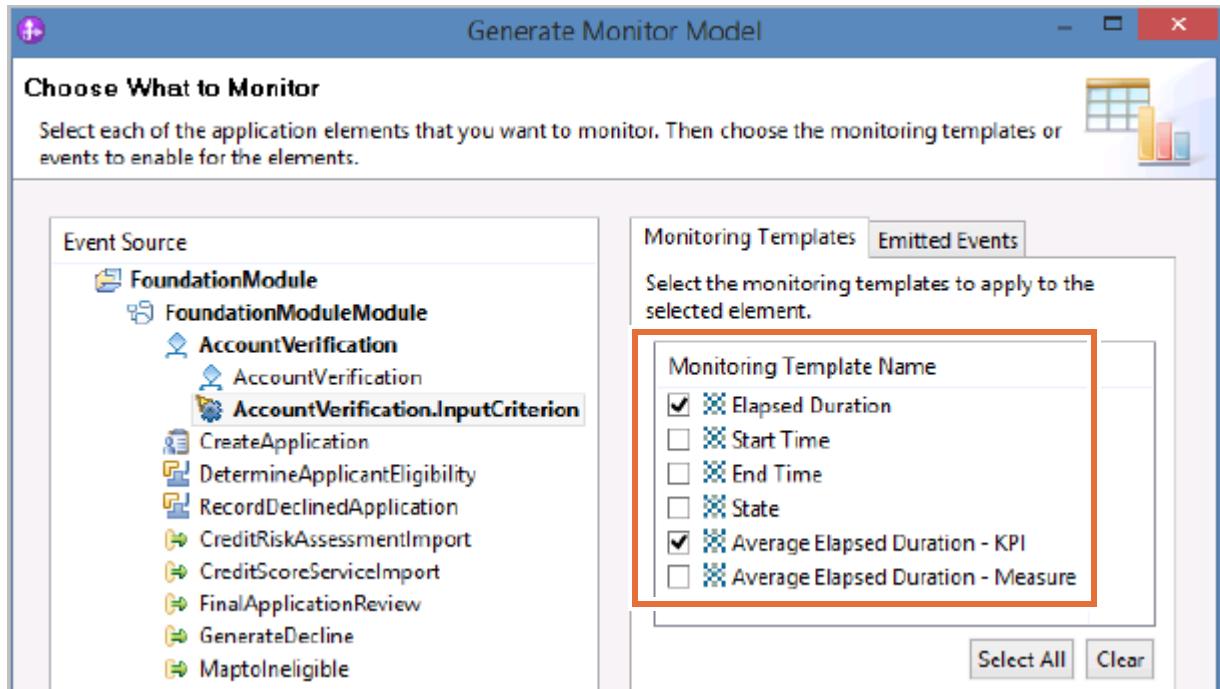
- b. Right-click the **FoundationModule** project and click **Generate Monitor Model**.



- c. In the **Generate Monitor Model** wizard, select **AccountVerificationMonitor** as the **Target Monitor Project** from the drop-down list, and keep the default monitor model name.

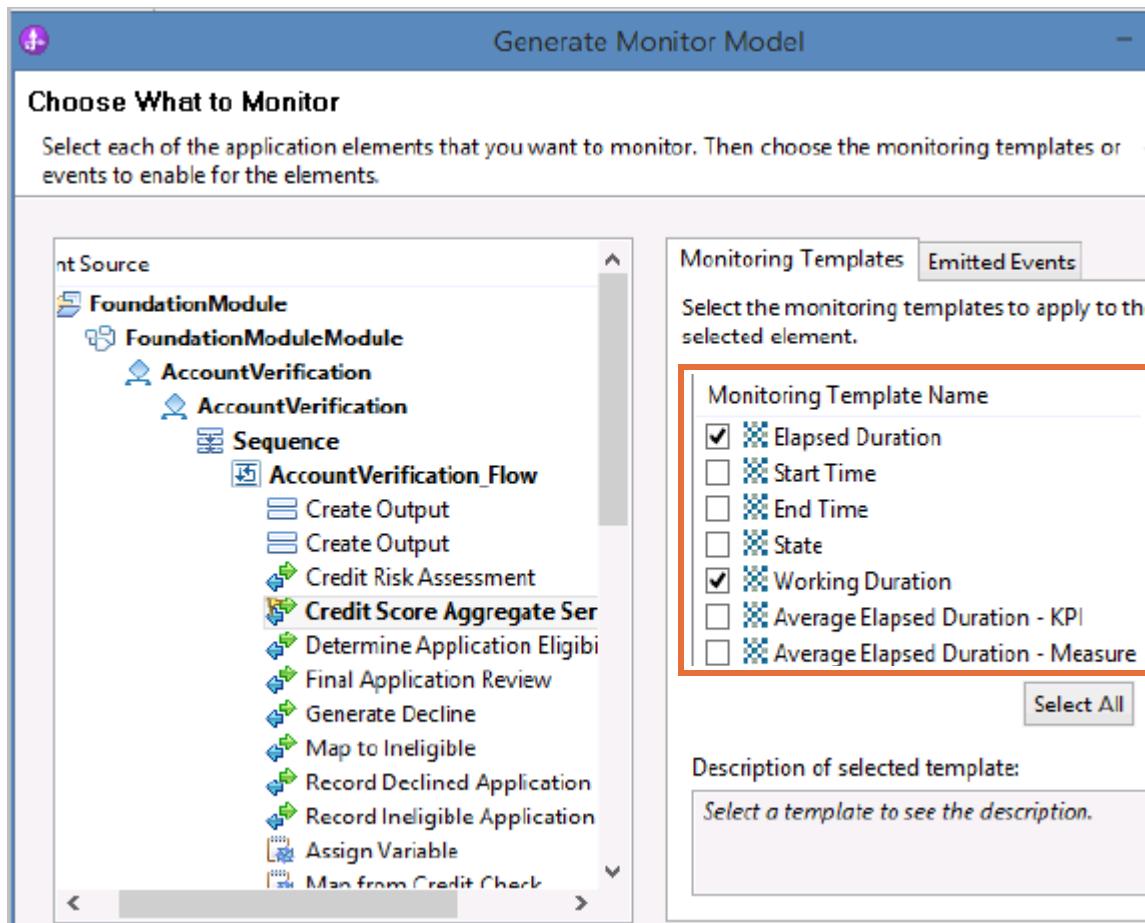


- d. Click **Next**.
- e. The next step of the generation process is to choose which application elements you want to monitor, and how you want to monitor them. Expand the **AccountVerification** BPEL process and select the **AccountVerification.InputCriterion** operation.
- f. In the **Monitoring Templates** tab, enable the **Elapsed Duration** and **Average Elapsed Duration - KPI** check boxes.



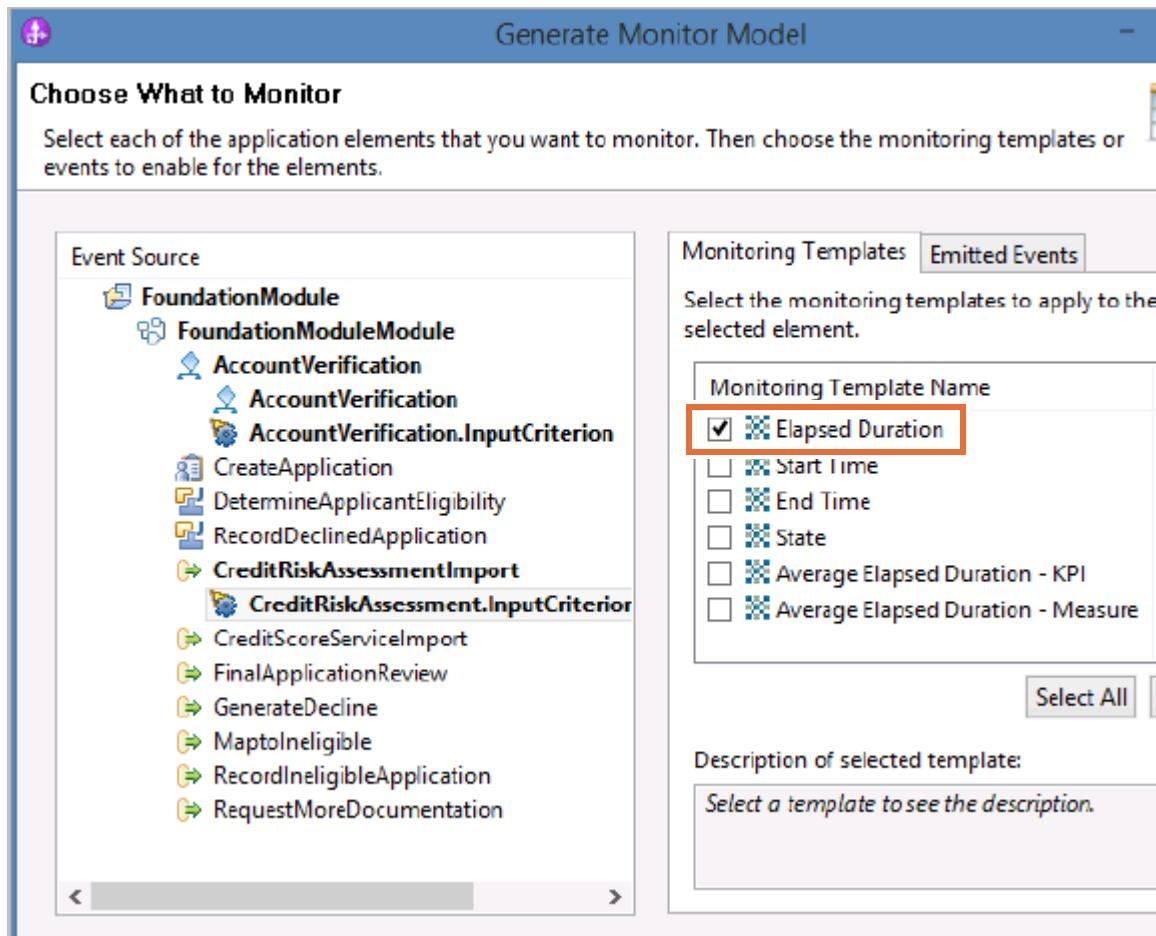
- g. Expand **AccountVerification > AccountVerification > Sequence > AccountVerification\_Flow** and select the **Credit Score Aggregate Service**. This service locates the credit score through a web service.

- \_\_ h. Check the **Elapsed Duration** and **Working Duration** check boxes.



- \_\_ i. Expand the **CreditRiskAssessmentImport** and select the **CreditRiskAssessment.InputCriterion** operation.

- \_\_ j. Check the **Elapsed Duration** check box.



- \_\_ k. Click **Next**.
- \_\_ l. After you select the artifacts to monitor, the next step is to select how they are monitored. By default, the two operations that you selected, **AccountVerification.InputCriterion** and **CreditRiskAssessment.InputCriterion**, and the **Credit Score Aggregate Service** invocation in the BPEL, are monitored according to **Event groups**.

If you have time, explore the settings, but do not change anything. Click **Next**.

Generate Monitor Model

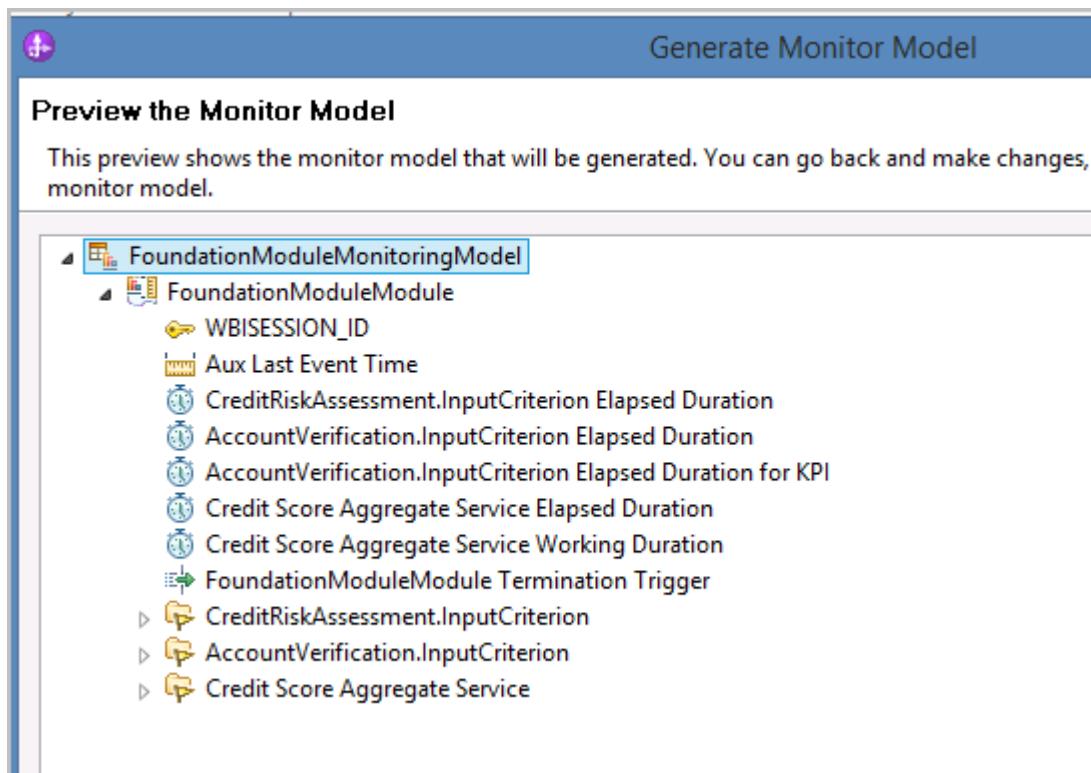
### Choose How to Monitor

The first column shows the elements that will be implemented in the monitor model. Click an element to see the emitted events.

| Event Source                        | Implementation     | Emitted Event Name                    |
|-------------------------------------|--------------------|---------------------------------------|
| FoundationModule                    | --                 | Credit Score Aggregate ServiceEN...   |
| FoundationModuleModule              | Monitoring context | Credit Score Aggregate ServiceEXIT... |
| AccountVerification                 | None               | Credit Score Aggregate ServiceEX...   |
| AccountVerification                 | None               | Credit Score Aggregate ServiceFAL...  |
| Sequence                            | None               | Credit Score Aggregate ServiceFC...   |
| AccountVerification_Flow            | None               | Credit Score Aggregate ServiceSKI...  |
| Credit Score Aggregate Service      | Event group        | Credit Score Aggregate ServiceSKI...  |
| AccountVerification.InputCriterion  | Event group        | Credit Score Aggregate ServiceST...   |
| CreditRiskAssessmentImport          | None               | Credit Score Aggregate ServiceTE...   |
| CreditRiskAssessment.InputCriterion | Event group        | Credit Score Aggregate ServiceFRE...  |

**Use Default Settings**

- \_\_\_ m. Expand **FoundationModuleMonitoringModel > FoundationModuleModule** and review the settings of the monitor model.



- \_\_\_ n. Click **Finish**.  
 \_\_\_ o. If the **Switch perspectives** window opens, click **Yes**.



### Information

The monitor model is built upon the event enablement, generation, and processing. The events are all in Common Base Event format. You can use the Monitor Model wizard and editor to enable events and establish generation templates for the purposes of monitoring your applications.

- \_\_\_ 6. Close the Monitor model editor by pressing **Ctrl+W**.

## **Part 2: Import and explore a Business Monitor model**

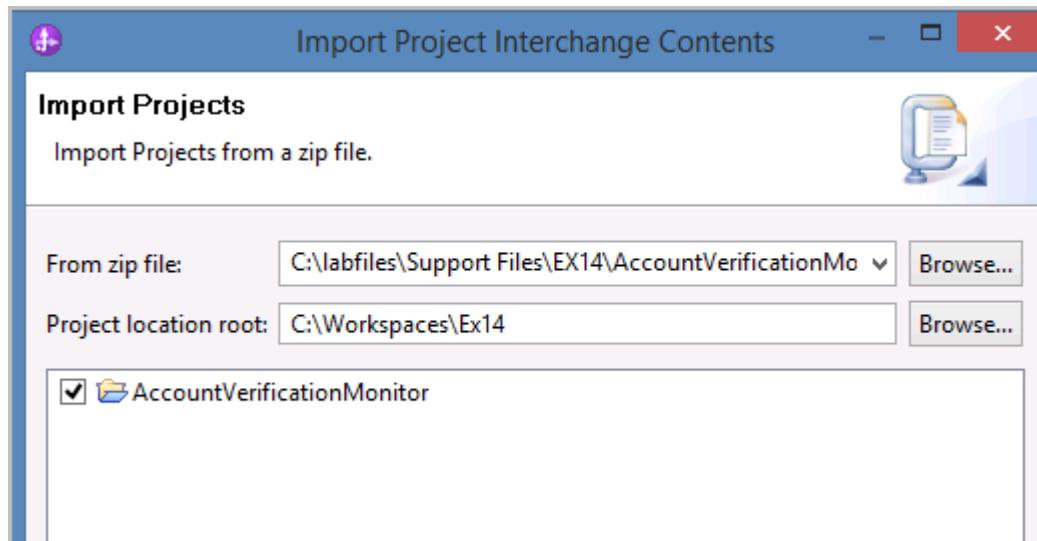


### Information

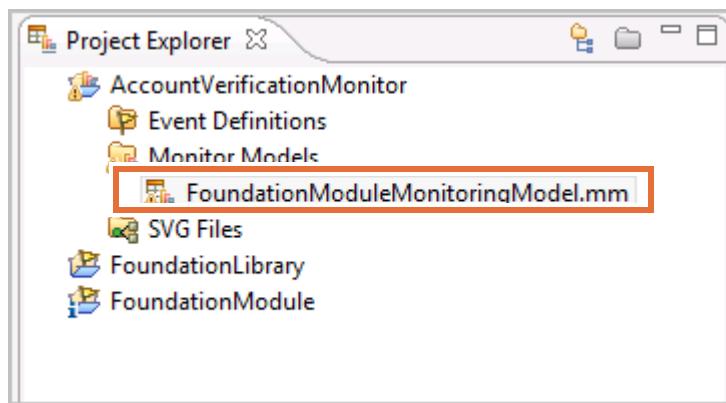
This section of the exercise is not intended to teach you how to create a complete monitor model with key performance indicators (KPIs), triggers, counters, or metrics. It is intended to demonstrate the tools that are available in IBM Integration Designer for automatically generating, and working with, monitor models.

In this section of the exercise, you import an existing monitor model that is based on the model that you previously generated. You then explore the elements of the model.

- 1. In IBM Integration Designer, click **File > Import**.
- 2. Click **Other > Project Interchange**.
- 3. Click **Next**.
- 4. Click **Browse** next to the **From zip file** field. Browse to **C:\labfiles\Support Files\EX14** and select **AccountVerificationMonitor.zip**. Click **Open**.

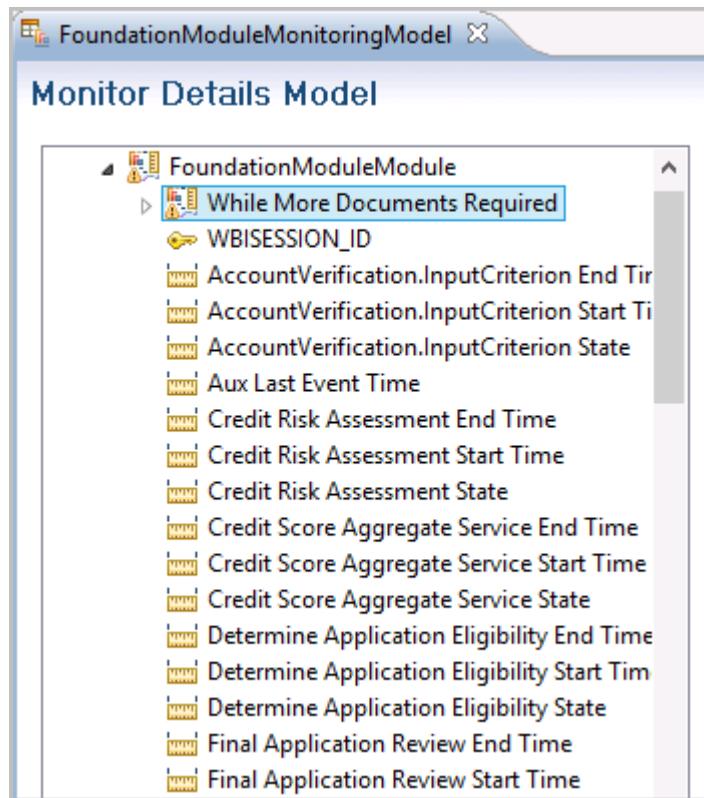


- 5. Click **Finish**.
- 6. On the **Confirm Overwrite** window, click **OK**. The import process overwrites the monitor model that you were building with one that is complete.
- 7. If you are not already in the Business Monitoring perspective, click **Window > Open Perspective > Other... > Business Monitoring**.
- 8. Expand **AccountVerificationMonitor > Monitor Models** and double-click **FoundationModuleMonitoringModel.mmm** to open it in the editor.



- 9. Explore the detailed contents of the monitor model.
  - a. In the **Monitor Details Model** tab, expand **FoundationModuleMonitoringModel > FoundationModuleModule > While More Documents Required**.

- \_\_\_ b. The imported monitor model contains many elements to monitor.



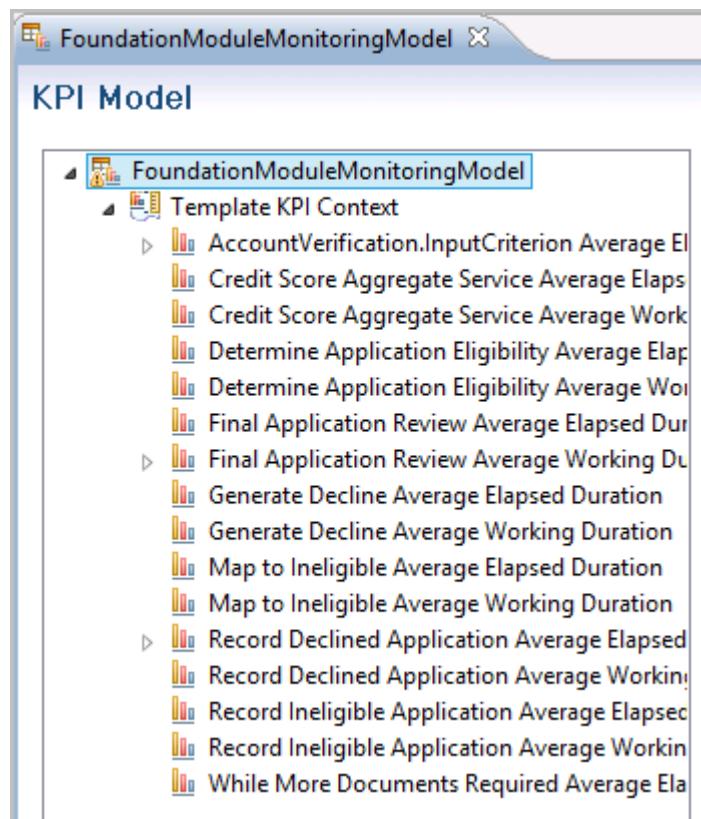
These elements include:

- 1) A **monitoring context**. A monitoring context defines all of the data to collect about an entity, such as a process or customer order.
- 2) A **key**. A key is a piece of information that characterizes and identifies the real-world entity that a monitoring context is tracking.
- 3) A **metric**. A metric is a holder for information, typically a business performance measurement.
- 4) A **trigger**. A trigger is a mechanism that detects an occurrence and can cause more processing in response.
- 5) An **inbound event**. A monitoring-enabled application generates a series of events. To indicate the events that are of interest to the monitoring context, you define inbound events in the model.
- 6) A **counter**. A counter is a specialized metric that counts occurrences, such as the number of ineligible applications, or iterations through a loop.
- 7) A **stopwatch**. A stopwatch is a specialized metric that tracks elapsed time.

- \_\_\_ 10. Explore the contents of the KPI model.
- \_\_\_ a. At the bottom of the Monitor model editor, click the **KPI Model** tab.



- \_\_ b. Expand **FoundationModuleMonitoringModel > Template KPI Context**.



- c. Click **AccountVerification.InputCriterion\_Average\_Elapsed\_Duration** and examine the key performance indicators, targets, and ranges on the right.
- **Key performance indicator (KPI):** KPIs are quantifiable measurements of the improvement or deterioration in the performance of an activity critical to the success of a business.
  - **Target:** A KPI can have a target, which is the exact value that the KPI can achieve.
  - **Range:** A KPI can also have ranges, each of which is a span of possible values.

**▼ KPI Details**

Edit the details of the KPI, which is a performance measurement used to track business objectives.

|              |                                                                                            |
|--------------|--------------------------------------------------------------------------------------------|
| ID:          | <input type="text" value="* AccountVerification.InputCriterion_Average_Elapsed_Duration"/> |
| Name:        | <input type="text" value="AccountVerification.InputCriterion Average Elapsed Duration"/>   |
| Description: | <input type="text" value=""/>                                                              |
| Type:        | <input type="text" value="Duration"/>                                                      |

|                                     |                                              |
|-------------------------------------|----------------------------------------------|
| <input type="checkbox"/>            | Hide from dashboards                         |
| <input checked="" type="checkbox"/> | Keep track of historical values for this KPI |

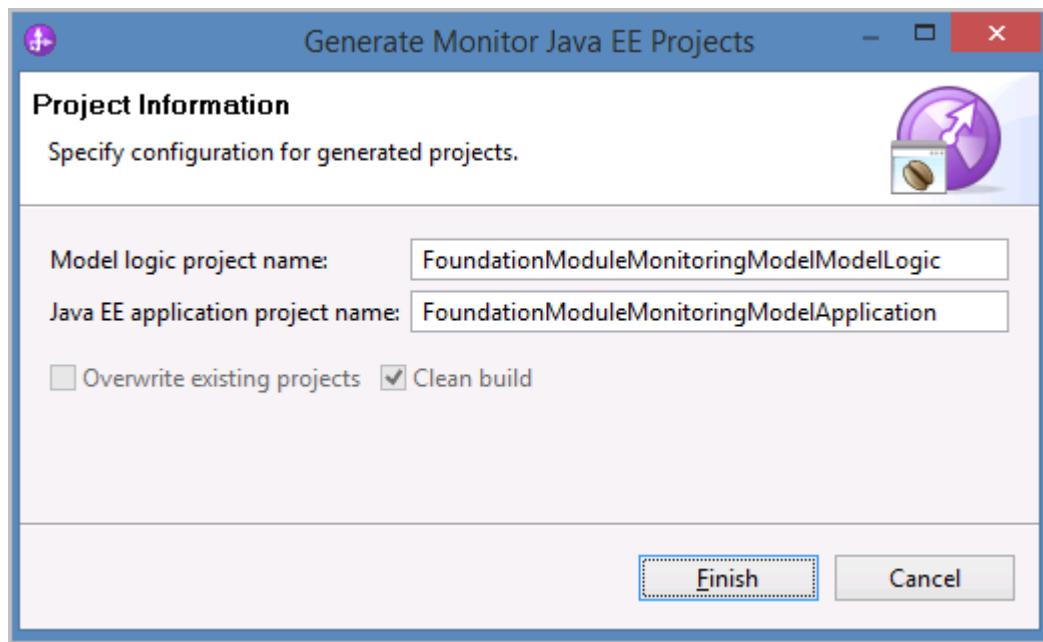
**▼ KPI Target and Ranges**

Specify a target, which is an exact value for the KPI to achieve, or ranges against which to track the KPI, or both.

|         |                                           |
|---------|-------------------------------------------|
| Target: | <input type="text" value="1 Days"/>       |
| Ranges: | <input type="text" value="Actual value"/> |

- 11. Close the Monitor model editor.
- 12. Generate a monitor Java project to associate with a process application.
- a. Right-click **FoundationModuleMonitoringModel.mmm** and select **Generate Monitor Java EE Projects**.

- \_\_\_ b. In the **Project Information** window, accept default values for the project names and click **Finish**.



- \_\_\_ 13. Minimize, but do not close, IBM Integration Designer.

### **Part 3: Associate a monitor model with a process application**

In this section of the exercise, you associate the monitor model with a process application in IBM Process Designer. A copy of the monitor model is kept on the IBM Process Center repository.



#### Troubleshooting

To effectively run this section of the exercise, it is important that only one server is running at a time. Before you begin these steps, make sure that IBM Process Server is not running.

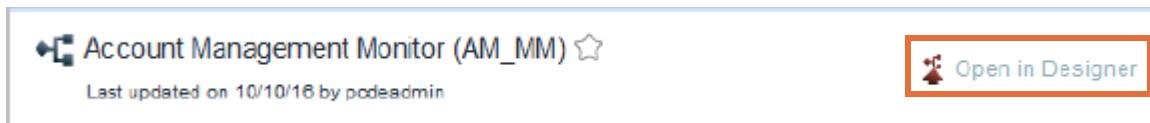
If you stopped the IBM Process Center Server in the previous exercise, you need to start it again. On the Windows desktop, do the following steps:

- \_\_\_ 1. Start the Process Center deployment manager.
  - \_\_\_ a. On your Windows desktop, select the shortcut that is titled: **Start Process Center deployment manager**. Double-click or press Enter to open the shortcut. It takes several minutes for deployment manager to start. When the deployment manager starts, you are prompted to press any key to continue. Press any key to close the command window.
  
- \_\_\_ 2. Start the Process Center node agent.
  - \_\_\_ a. On your Windows desktop, select the shortcut that is titled: **Start Process Center node agent**. Double-click or press Enter to open the shortcut. It takes several minutes for node agent to start. When the node agent starts, you are prompted to press any key to continue. Press any key to close the command window.

- \_\_\_ 3. Start the Process Center Cluster.
  - \_\_\_ a. On your Windows desktop, select the shortcut that is titled: **Start Process Center Cluster**. Double-click or press Enter to open the shortcut. It takes several minutes (longer than the deployment manager and node agent processes) for the cluster to start. When the cluster starts, you are prompted to press any key to continue. Press any key to close the command window.
  
- \_\_\_ 4. After the IBM Process Center cluster is running, start IBM Process Designer, *if* you stopped it in the previous exercise.
  - \_\_\_ a. On the Windows desktop, double-click the shortcut titled **Start IBM Process Designer**. It takes several moments for the application to open.
  - \_\_\_ b. Use `pcdeadmin` in the **User Name** field and `web1sphere` in the **Password** field and click **Login**.
  - \_\_\_ c. If you are prompted with security alerts, then click **Yes**.
  
- \_\_\_ 5. If you are not already in the Process Center perspective, click the Process Center icon in the upper right to return to the perspective.



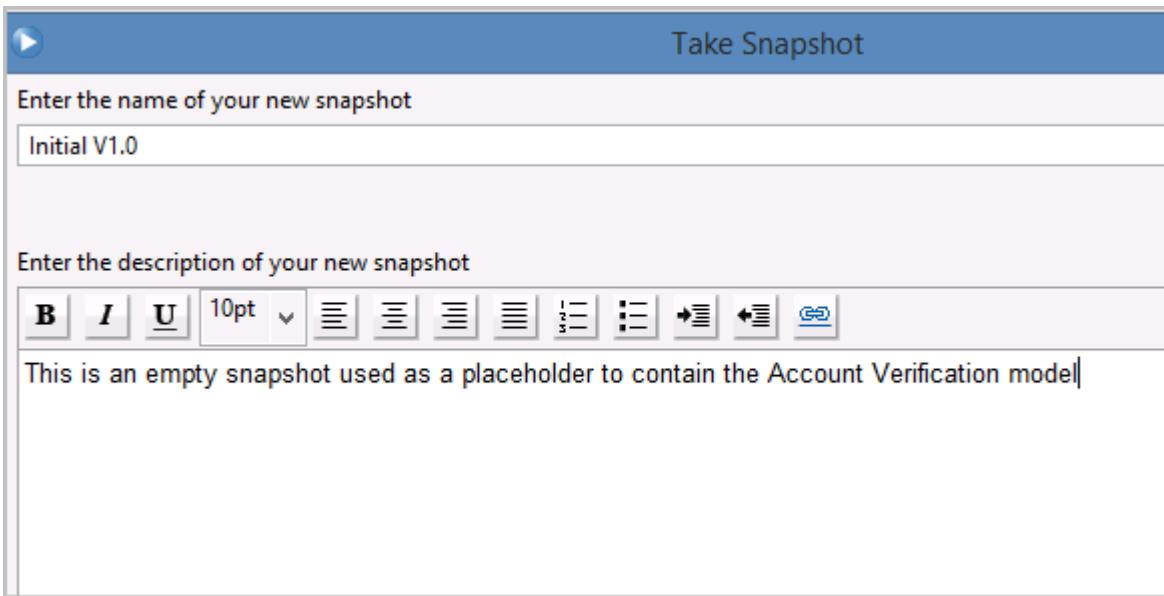
- \_\_\_ 6. Create a process application to contain the monitor model.
  - \_\_\_ a. In the **Process Apps** tab, click **Create New Process App**.
  - \_\_\_ b. Set the **Process App Name** to: Account Management Monitor
  - \_\_\_ c. Set the **Acronym** to: `AM_MM`
  - \_\_\_ d. If you want, add a meaningful description.
  - \_\_\_ e. Click **Create**.
  
- \_\_\_ 7. Take a snapshot of the empty process application.
  - \_\_\_ a. In the **Process Apps** tab, click the **Open in Designer** link next to **Account Management Monitor**.



- \_\_\_ b. Click **Snapshot**.



- \_\_\_ c. Set the name of the snapshot to: Initial v1.0  
If you want, add a meaningful description.



- \_\_\_ d. Click **OK**.
- \_\_\_ e. Return to the Process Center perspective in IBM Process Designer by clicking the **Process Center** icon at the upper-right corner.
- \_\_\_ 8. Restore IBM Integration Designer, but do not close IBM Process Designer.
- \_\_\_ 9. Associate the monitor model with the empty process application.



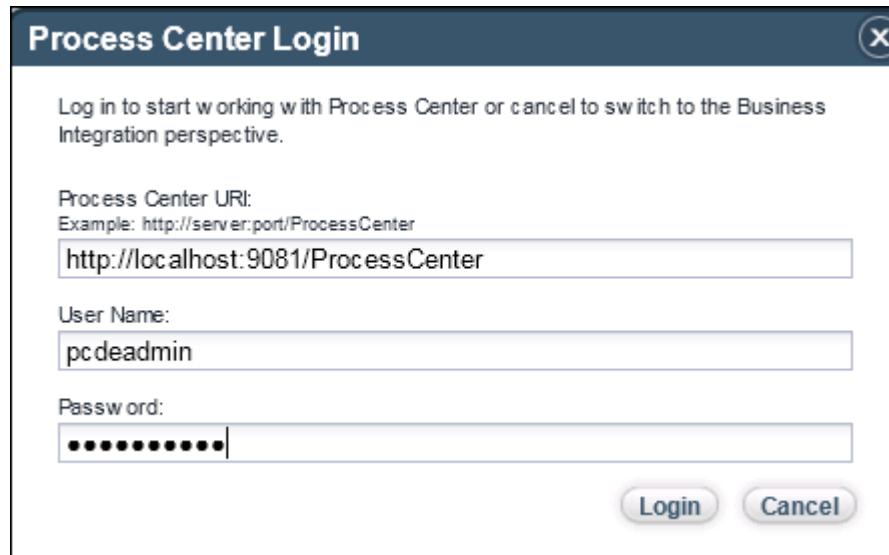
### Attention

When you associate a monitor model with a process application, it is a good practice to associate it with an empty or new process application. If the monitor model exists in the same process application as commonly used business processes, then every time the process app is updated, multiple snapshots contain the same monitor model.

- 
- \_\_\_ a. Click **Window > Open Perspective > Other**.
  - \_\_\_ b. Select **Process Center** and click **OK**.

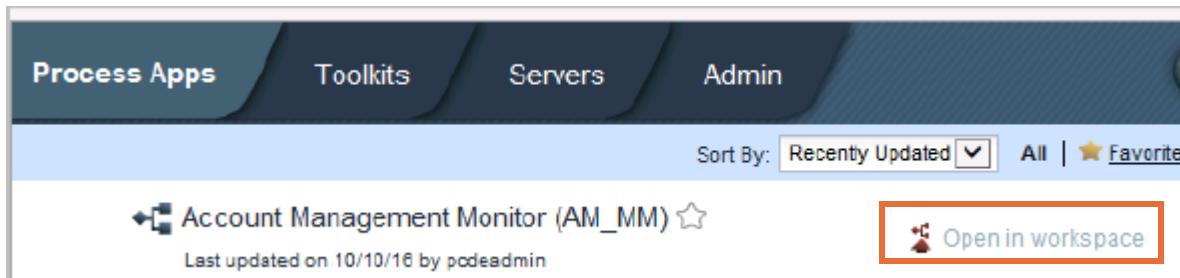
\_\_\_ c. In the Process Center Login window, enter the following information:

- **Process Center URI:** `http://localhost:9081/ProcessCenter`
- **User Name:** `pcdeadmin`
- **Password:** `web1sphere`



\_\_\_ d. Click **Login**.

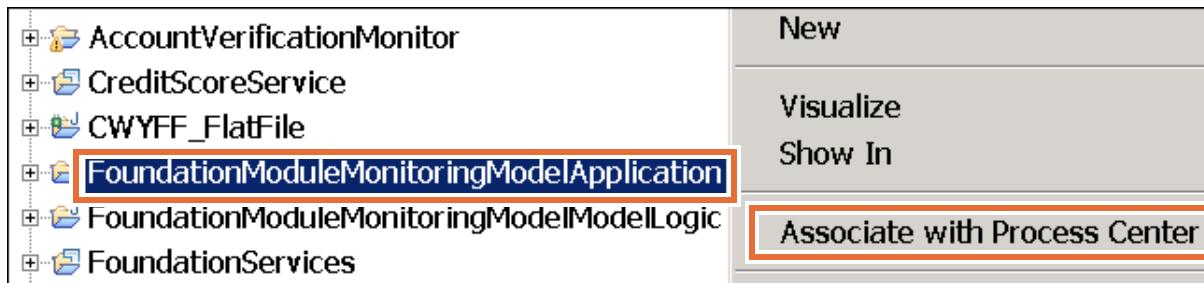
- \_\_\_ e. If you are prompted with **Secure Storage** window, click **Cancel**, and click **OK** in the **Security Storage Warning** window.
- \_\_\_ f. Click **Yes** when security alert messages are displayed.
- \_\_\_ g. Close the **Getting Started** window.
- \_\_\_ h. In the **Process Apps** tab, click the **Open in Workspace** link next to the **Account Management Monitor** process app.



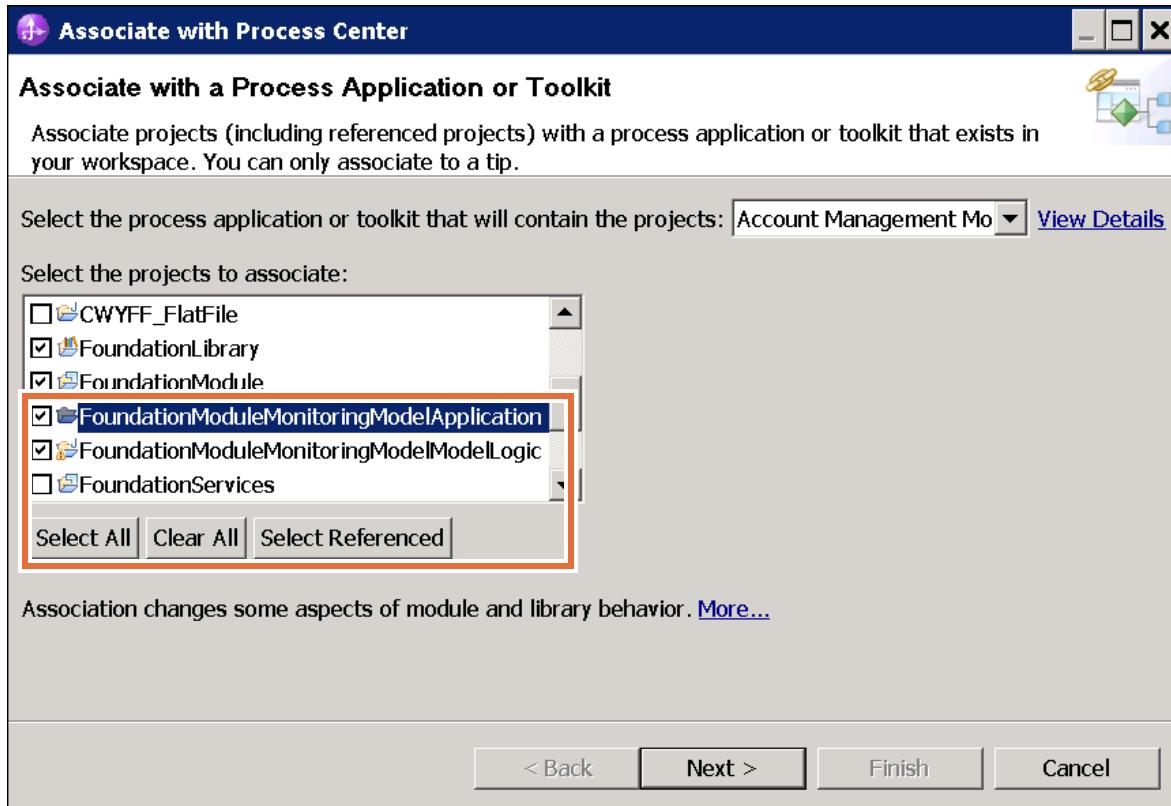
It takes a few moments to create the artifacts for the process app in IBM Integration Designer.

- \_\_\_ i. In the **Open Process Applications and Toolkits in Workspace** window, click **OK**.

- \_\_ j. Right-click the `FoundationModuleMonitoringModelApplication` project and click **Associate with Process Center**.



- \_\_ k. In the **Associate with Process Center** window, set the process application to: Account Management Monitor
- \_\_ l. From the list of projects to associate, click `FoundationModuleMonitoringModelApplication`, `FoundationModuleMonitoringModelModelLogic`, `FoundationLibrary`, and `FoundationModule` to select them.



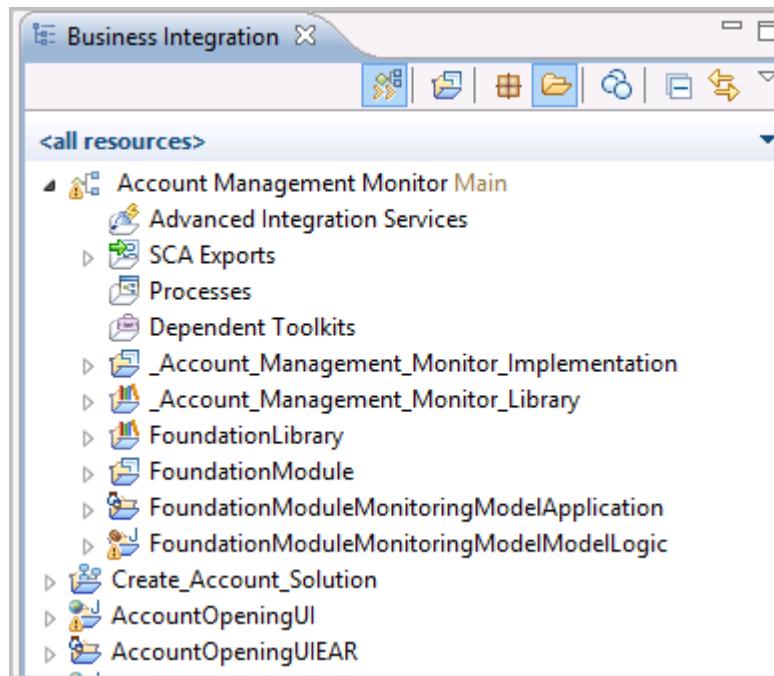
- \_\_ m. Click **Next**.
- \_\_ n. Click **Finish**.



### Attention

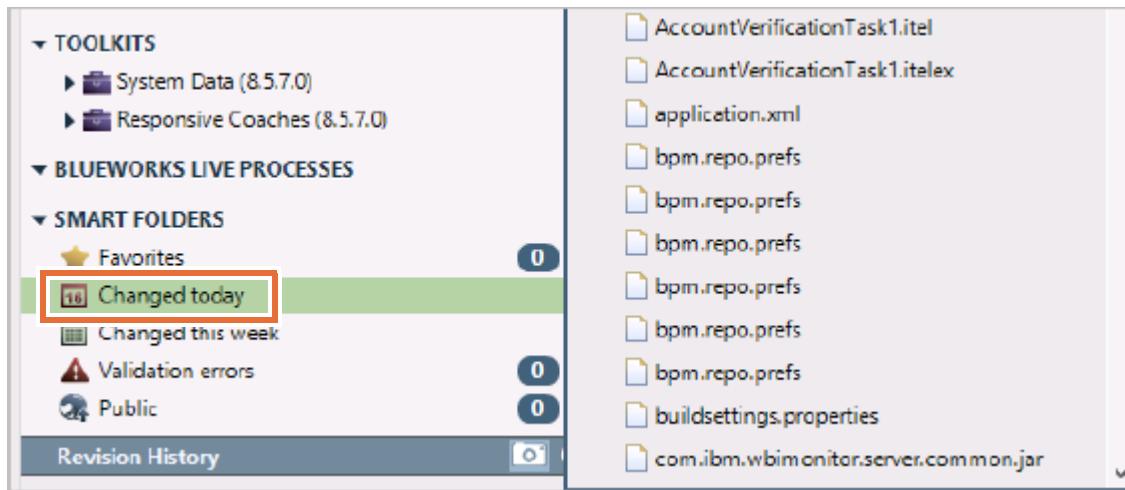
Having a process app with multiple snapshots of the same monitor model becomes a problem at run time when you deploy all these snapshots on the server. You cannot deploy the same monitor model version to the IBM Business Monitor if that version is already deployed. A warning message is shown when you associate a monitor project to a process application.

- \_\_\_ o. If you get the **Switch Business Integration View to Detailed Mode** window, click **Yes**.
- \_\_\_ 10. In the Business Integration perspective of IBM Integration Designer, expand the **Account Management Monitor** project. It contains the monitor project and its associated business integration module and library.



- \_\_\_ 11. Right-click **Account Management Monitor Main** and click **Publish to Process Center**.
- \_\_\_ 12. Verify the association to the process application.
  - \_\_\_ a. Restore IBM Process Designer.
  - \_\_\_ b. In the **Process Apps** tab, click the **Open in Designer** link next to the **Account Management Monitor** process application.

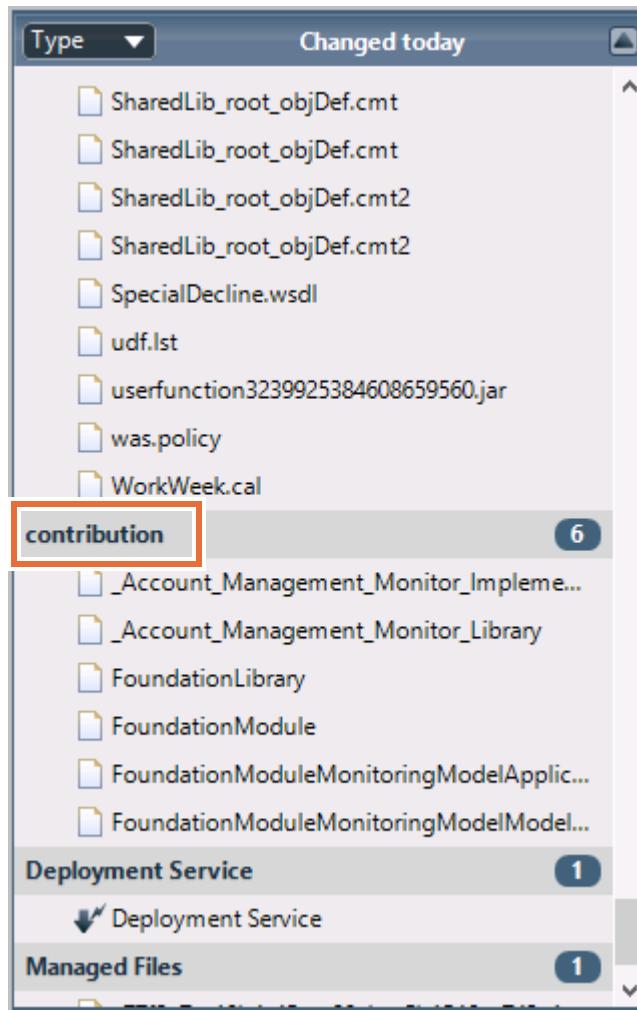
- c. In the Designer view, click **Changed today**. Because the monitor assets are created in IBM Integration Designer, the assets are not shown in IBM Process Designer for editing. However, the change window demonstrates that the assets are added to the process app.



### Information

Any IBM Integration Designer asset that has the file extension .mon contains events for IBM Business Monitor.

- \_\_ d. Scroll to the bottom of the **Changed today** list. The **contribution** list contains a list of all project assets that are added to the process application.



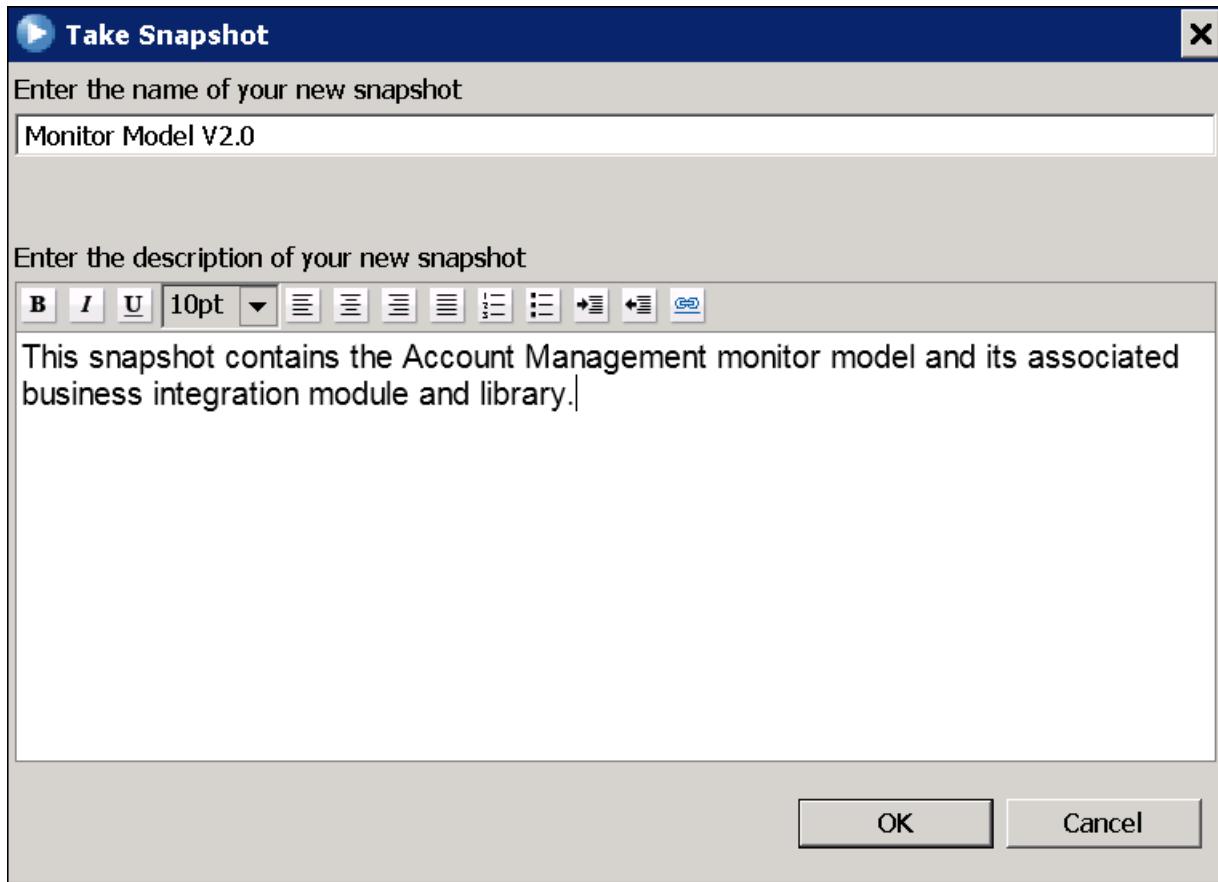
- \_\_ 13. Take a snapshot of the process app.

- \_\_ a. Click the **Snapshot** tool.



- \_\_ b. Set the name of the snapshot to: Monitor Model v2.0

- \_\_\_ c. If you want, add a meaningful description.



- \_\_\_ d. Click **OK**.
- \_\_\_ 14. Return to the Process Center perspective in IBM Process Designer, but do not close or exit IBM Process Designer. If you are prompted with the **Error: Connection to IBM BPM Server is lost**, ignore the error and click **Close**.



- \_\_\_ 15. Close the IBM Integration Designer. Click **File > Exit**, or click **X** in the upper-right corner.

**End of exercise**



IBM Training



© Copyright International Business Machines Corporation 2017.