# Contents

<table>
<thead>
<tr>
<th>Preface</th>
<th>xi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audience</td>
<td>xi</td>
</tr>
<tr>
<td>Documentation Accessibility</td>
<td>xi</td>
</tr>
<tr>
<td>Related Documents</td>
<td>xii</td>
</tr>
<tr>
<td>Conventions</td>
<td>xii</td>
</tr>
</tbody>
</table>

## 1 Introduction to the SOA Sample Application
1.1 Introduction to Fusion Order Demo | 1-1 |
1.2 Setting Up the Development Environment for Creating the WebLogic Fusion Order Demo Application | 1-2 |
1.2.1 Task 1: Install Oracle JDeveloper Studio | 1-2 |
1.2.2 Task 2: Install the Fusion Order Demo Application | 1-2 |
1.2.3 Task 3: Install Oracle SOA Suite | 1-2 |
1.2.4 Task 4: Create a Connection to an Oracle WebLogic Server | 1-5 |
1.3 Viewing the WebLogicFusionOrderDemo Application Artifacts | 1-6 |
1.4 Understanding the OrderBookingComposite Flow | 1-7 |

## 2 Running the Sample Application
2.1 Deploying the Fusion Order Demo Applications | 2-1 |
2.1.1 Task 1: Install the Schema for the Fusion Order Demo Application | 2-1 |
2.1.2 Task 2: Deploy the Store Front Module | 2-2 |
2.1.3 Task 3: Deploy the WebLogic Fusion Order Demo Application | 2-4 |
2.2 Placing Two Orders in the Store Front | 2-6 |
2.3 Starting Fusion Middleware Control to Monitor Orders | 2-10 |
2.4 Monitoring the First Order | 2-10 |
2.5 Monitoring the Second Order | 2-18 |
2.5.1 Task 1: View the Order in the Fusion Middleware Control | 2-18 |
2.5.2 Task 2: Use the Oracle BPM Worklist to Approve the Order | 2-21 |
2.5.3 Task 3: View the Approval in the Fusion Middleware Control | 2-21 |
2.6 Undeploying the Composites for the WebLogic Fusion Order Demo Application | 2-23 |

## 3 Creating the SOA Application
3.1 About the PartnerSupplierComposite Composite | 3-1 |
3.2 Creating the WebLogicFusionOrderDemo Application and the PartnerSupplierComposite Composite | 3-2 |
3.2.1 Task 1: Create the WebLogicFusionOrderDemo Application and the PartnerSupplierComposite Composite 3-2
3.2.2 Task 2: Create the ExternalPartnerSupplier BPEL Process 3-3
3.2.3 Task 3: Modify the ExternalPartnerSupplier BPEL Process 3-6
3.2.4 Task 4: Deploy the PartnerSupplierComposite Composite 3-9
3.2.5 Task 6: Initiate a Test Instance for the PartnerSupplierComposite Composite 3-9

4 Creating the OrderBookingComposite Composite
4.1 About the OrderBookingComposite Composite 4-1
4.2 Approaches for Creating OrderBookingComposite 4-4
4.3 Creating the OrderBookingComposite Project 4-5
4.3.1 Task 1: Create the OrderBookingComposite Project 4-5
4.3.2 Task 2: Create the OrderProcessor BPEL Process 4-5
4.3.3 Task 3: Add the ADF Business Components Service Runtime Library 4-8
4.4 About the OrderProcessor Process 4-8

5 Creating the First Half of the OrderProcessor BPEL Process
5.1 Overview of Tasks for Creating the First Half of OrderProcessor 5-1
5.2 Copying Services Used by the OrderProcessor BPEL Process 5-2
5.3 Adding the StoreFrontService Service 5-3
5.3.1 Task 1: Copy the WSDL Needed for StoreFrontService 5-3
5.3.2 Task 2: Create a Web Service for StoreFrontService 5-3
5.4 Wiring the OrderProcessor BPEL Process to the StoreFrontService Service 5-5
5.5 Creating the gOrderInfoVariable Variable 5-6
5.6 Creating the Scope_RetrieveOrder Scope 5-8
5.6.1 Task 1: Add the Scope_RetrieveOrder Scope 5-9
5.6.2 Task 2: Create findOrderId Bind Entity Activity 5-9
5.7 Creating the Scope_RetrieveCustomerForOrder Scope 5-11
5.7.1 Task 1: Add the Scope_RetrieveCustomerForOrder Scope 5-12
5.7.2 Task 2: Create the InvokeCustomerService Activity 5-12
5.7.3 Task 3: Create the AssignCustomerId Activity 5-14
5.7.4 Task 4: Deploy the OrderBookingComposite Composite 5-15
5.7.5 Task 5: Deploy the OrderSDOComposite Composite 5-16
5.7.6 Task 6: Initiate a Test Instance for the OrderBookingComposite Composite 5-16
5.8 Creating CreditCardAuthorizationService Service 5-17
5.8.1 Task 1: Copy WSDL File Needed for CreditCardAuthorizationService 5-17
5.8.2 Task 2: Create a Web Service for CreditCardAuthorizationService 5-17
5.9 Creating the Scope_AuthorizeCreditCard Scope 5-18
5.9.1 Task 1: Add the Scope_AuthorizeCreditCard Scope 5-19
5.9.2 Task 2: Create the InvokeCheckCredit Invoke Activity 5-19
5.9.3 Task 3: Create the Assign_CreditCheckInput Activity 5-20
5.9.4 Task 4: Create Switch Activity 5-23
5.10 Creating Catch Branches for the Scope_AuthorizeCreditCard 5-24
5.10.1 Task 1: Modify the OrderProcessor.wsdl File for the gOrderProcessorFaultVariable Variable 5-25
5.10.2 Task 2: Create the gOrderProcessorFaultVariable Variable 5-28
5.10.3 Task 3: Add Catch Branches to the Scope_AuthorizeCreditCard 5-29
### 6 Creating the Second Half of the OrderProcessor BPEL Process

#### 6.1 Overview of Tasks for Creating the Second Half of OrderProcessor

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.11.1</td>
<td>Task 1: Create Scope_CheckApprovalLimit Scope</td>
</tr>
<tr>
<td>5.11.2</td>
<td>Task 2: Add the IOrderApproved Variable</td>
</tr>
<tr>
<td>5.11.3</td>
<td>Task 3: Create the Assign_DefaultNotRequiresApproval Assign Activity</td>
</tr>
<tr>
<td>5.11.4</td>
<td>Task 4: Create the RequiresApprovalRule Business Rule</td>
</tr>
<tr>
<td>5.11.5</td>
<td>Task 5: Reference the RequiresApprovalRule Dictionary in the BPEL Designer</td>
</tr>
<tr>
<td>5.11.6</td>
<td>Task 6: Define a Variable in Rules Designer</td>
</tr>
<tr>
<td>5.11.7</td>
<td>Task 7: Add a New Rule for the Ruleset in Rules Designer</td>
</tr>
<tr>
<td>5.11.8</td>
<td>Task 8: Redeploy the OrderBookingComposite Composite</td>
</tr>
<tr>
<td>5.11.9</td>
<td>Task 9: Initiate a Test Instance for the OrderBookingComposite Composite</td>
</tr>
<tr>
<td>5.12</td>
<td>Task 6: Initiate a Test Instance for the OrderBookingComposite Composite</td>
</tr>
</tbody>
</table>

#### 6.2 Creating the Scope_RetrieveQuotes Flow

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.2.1</td>
<td>Task 1: Add the Scope_RetrieveQuotes Scope</td>
</tr>
<tr>
<td>6.2.2</td>
<td>Task 2: Create the InternalWarehouseService BPEL Process</td>
</tr>
<tr>
<td>6.2.3</td>
<td>Task 3: Modify the InternalWarehouseService Process</td>
</tr>
<tr>
<td>6.2.4</td>
<td>Task 4: Wire OrderProcessor to the InternalWarehouseService Process</td>
</tr>
<tr>
<td>6.2.5</td>
<td>Task 5: Create the PartnerSupplierService Service</td>
</tr>
<tr>
<td>6.2.6</td>
<td>Task 6: Create a PartnerSupplier Mediator Service for the PartnerSupplierService Service</td>
</tr>
<tr>
<td>6.2.7</td>
<td>Task 7: Create Routing Rules Between the PartnerSupplierMediator Mediator to the ExternalPartnerSupplier Service</td>
</tr>
<tr>
<td>6.2.8</td>
<td>Task 8: Wire OrderProcessor to the PartnerSupplierMediator Mediator</td>
</tr>
<tr>
<td>6.2.9</td>
<td>Task 9: Add the gWarehouseQuotes Variable</td>
</tr>
<tr>
<td>6.2.10</td>
<td>Task 10: Add Activities to Obtain a Quote from the InternalWarehouse Process</td>
</tr>
<tr>
<td>6.2.11</td>
<td>Task 11: Add Activities to Obtain a Quote from the PartnerSupplierMediator Mediator</td>
</tr>
</tbody>
</table>

#### 6.3 Creating the Scope_SelectPreferredSupplier Scope

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.3.1</td>
<td>Task 1: Create the Scope_SelectPreferredSupplier Scope</td>
</tr>
<tr>
<td>6.3.2</td>
<td>Task 2: Add the gPreferredSupplier Variable</td>
</tr>
<tr>
<td>6.3.3</td>
<td>Task 3: Create the EvaluatePreferredSupplierRule Business Rule</td>
</tr>
<tr>
<td>6.3.4</td>
<td>Task 4: Reference the RequiresApprovalRule Dictionary in the BPEL Designer</td>
</tr>
<tr>
<td>6.3.5</td>
<td>Task 5: Add a New Rule for Ruleset in Rules Designer</td>
</tr>
</tbody>
</table>

#### 6.4 Creating the Services and Routing Required for the Scope_FulfillOrder Scope

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.4.1</td>
<td>Task 1: Create USPSShipment File Adapter</td>
</tr>
<tr>
<td>6.4.2</td>
<td>Task 2: Create FulfillmentBatch JMS Service</td>
</tr>
<tr>
<td>6.4.3</td>
<td>Task 3: Create FulfillOrder Mediator Service Component</td>
</tr>
<tr>
<td>6.4.4</td>
<td>Task 4: Reference the RequiresApprovalRule Dictionary in the BPEL Designer</td>
</tr>
<tr>
<td>6.4.5</td>
<td>Task 5: Redeploy the FulfillmentComposite Composite</td>
</tr>
<tr>
<td>6.4.6</td>
<td>Task 6: Initiate a Test Instance for the FulfillmentComposite Composite</td>
</tr>
</tbody>
</table>

#### 6.5 Creating the Services and Routing Required for the Scope_FulfillmentServiceScope

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.5.1</td>
<td>Task 1: Create the FulfillmentService Service</td>
</tr>
<tr>
<td>6.5.2</td>
<td>Task 2: Add the gFulfillmentVariable</td>
</tr>
<tr>
<td>6.5.3</td>
<td>Task 3: Create the FulfillmentOutcome Mediator Service for the FulfillmentService Service</td>
</tr>
<tr>
<td>6.5.4</td>
<td>Task 4: Reference the RequiresApprovalRule Dictionary in the BPEL Designer</td>
</tr>
<tr>
<td>6.5.5</td>
<td>Task 5: Redeploy the FulfillmentComposite Composite</td>
</tr>
<tr>
<td>6.5.6</td>
<td>Task 6: Initiate a Test Instance for the FulfillmentComposite Composite</td>
</tr>
</tbody>
</table>
6.4.4 Task 4: Create Routing Rules ................................................................. 6-38
6.4.5 Task 5: Wire OrderProcessor to FulfillOrder Mediator Service................. 6-40
6.5 Creating the Scope_FulfillOrder Scope....................................................... 6-41
6.5.1 Task 1: Add the Scope_FulfillOrder Scope .............................................. 6-41
6.5.2 Task 2: Create the InvokeFulfillOrder Activity ...................................... 6-41
6.5.3 Task 3: Create the AssignFulfillRequest Activity .................................. 6-43
6.6 Creating the Scope_UpdateStatusToComplete Scope for Completed Orders .... 6-43
6.7 Creating the Scope_NotifyCustomerofCompletion Scope ............................ 6-45
6.8 Adding a Catch Branch for Incomplete Orders for the Entire Process ............ 6-47

7 Adding the OrderPendingEvent Mediator Service Component
7.1 Task 1: Create the NewOrderSubmitted Business Event ............................. 7-1
7.2 Task 2: Create Mediator Service Component to Subscribe to NewOrderSubmitted Business Event .............................................................. 7-2
7.3 Task 3: Route OrderPendingEvent Mediator Service Component to OrderProcessor BPEL Process ............................................................. 7-3

8 Adding a Flow to Update Orders
8.1 Task 1: Copy Schema File Needed for Business Event ............................... 8-1
8.2 Task 2: Create the UpdateOrderStatus Mediator Service Component to Publish the OrderUpdateEvent Business Event ........................................ 8-1
8.3 Task 3: Create a Routing Rule to Initiate the Business Event ...................... 8-2
8.4 Task 4: Create the OrderUpdateEventMediator Mediator Service Component to Subscribe to the OrderUpdateEvent Business Event ...................... 8-4
8.5 Task 5: Create a Routing Rule to Send Order Updates to the StoreFrontService service. 8-5
8.6 Task 6: Redeploy the OrderBookingComposite Composite ........................ 8-7
8.7 Task 7: Initiate a Test Instance for the OrderBookingComposite Composite .... 8-8

9 Creating the Task Display Form for the ApprovalHumanTask Human Task
9.1 About the Task Form...................................................................................... 9-1
9.2 Task 1: Create a New Task Form for the ApprovalHumanTask Human Task .... 9-2
9.3 Task 2: Add the ADF Business Components Service Runtime Library to the Project .... 9-3
9.4 Task 3: Create the Contents for the Task Form ........................................... 9-3
9.5 Task 4: Deploy the OrderApprovalHumanTask Task Form .......................... 9-7

10 Conclusion

A Create the JMS Topic for the FulfillmentBatch Adapter
A.1 Task 1: Create the JMS Topic....................................................................... A-1
A.2 Task 2: Create the JMS Topic Connection Factory ..................................... A-2
A.3 Task 3: Add the Connection Pool ................................................................. A-3

B ant Scripts
B.1 About ant Scripts.......................................................................................... B-1
Index
Welcome to the Oracle Fusion Middleware Tutorial for Running and Building an Application with Oracle SOA Suite.

**Audience**

This tutorial is intended for developers to create and deploy an SOA composite application using Oracle SOA Suite.

**Documentation Accessibility**

Our goal is to make Oracle products, services, and supporting documentation accessible to all users, including users that are disabled. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at [http://www.oracle.com/accessibility/](http://www.oracle.com/accessibility/).

**Accessibility of Code Examples in Documentation**

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

**Accessibility of Links to External Web Sites in Documentation**

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

**Deaf/Hard of Hearing Access to Oracle Support Services**

To reach Oracle Support Services, use a telecommunications relay service (TRS) to call Oracle Support at 1.800.223.1711. An Oracle Support Services engineer will handle technical issues and provide customer support according to the Oracle service request process. Information about TRS is available at [http://www.fcc.gov/cgb/consumerfacts/trs.html](http://www.fcc.gov/cgb/consumerfacts/trs.html), and a list of phone numbers is available at [http://www.fcc.gov/cgb/dro/trsphonebk.html](http://www.fcc.gov/cgb/dro/trsphonebk.html).
Related Documents

For more information, see the following documents:

- Oracle Fusion Middleware Getting Started with Oracle SOA Suite
- Oracle Fusion Middleware Developer’s Guide for Oracle SOA Suite
- Oracle Fusion Middleware Administrator’s Guide for Oracle SOA Suite

Conventions

The following text conventions are used in this document:

<table>
<thead>
<tr>
<th>Convention</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>boldface</td>
<td>Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.</td>
</tr>
<tr>
<td>italic</td>
<td>Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.</td>
</tr>
<tr>
<td>monospace</td>
<td>Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.</td>
</tr>
</tbody>
</table>
Introduction to the SOA Sample Application

The WebLogic Fusion Order Demo application of the Fusion Order Demo demonstrates using Oracle SOA Suite for processing orders from a Web shopping storefront. This tutorial focuses on running and building the WebLogic Fusion Order Demo application.

This chapter contains the following sections:

- Section 1.1, "Introduction to Fusion Order Demo"
- Section 1.2, "Setting Up the Development Environment for Creating the WebLogic Fusion Order Demo Application"
- Section 1.3, "Viewing the WebLogicFusionOrderDemo Application Artifacts"
- Section 1.4, "Understanding the OrderBookingComposite Flow"

For an overview of Oracle SOA Suite, see Oracle Fusion Middleware Getting Started with Oracle SOA Suite.

1.1 Introduction to Fusion Order Demo

Run by a fictitious company called Global Company, the Fusion Order Demo provides two main parts, the Store Front module and the WebLogic Fusion Order Demo application.

The StoreFront module sells electronic devices through a storefront-type Web application.

The StoreFront module contains the following projects:

- **StoreFrontService**: This project provides access to the storefront data and provides transaction support to update data for customers, orders, and products.
- **StoreFrontUI**: This project provides Web pages that the customer uses to browse the storefront, place orders, register on the site, view order information, and update the user profile.

The StoreFrontUI project uses JavaServer Faces (JSF) as the view technology, and relies on the Oracle ADF Model layer to interact with Oracle Application Development Framework (Oracle ADF) Business Components in the StoreFrontService project.

For a detailed description of the StoreFront module and its projects, see the Oracle Fusion Middleware Fusion Developer’s Guide for Oracle Application Development Framework.

Once a customer places an order, the WebLogic Fusion Order Demo application processes the order.
1.2 Setting Up the Development Environment for Creating the WebLogic Fusion Order Demo Application

To prepare for the creation of the WebLogic Fusion Order Demo application, complete the following tasks.

- **Task 1: Install Oracle JDeveloper Studio**
- **Task 2: Install the Fusion Order Demo Application**
- **Task 3: Install Oracle SOA Suite**
- **Task 4: Create a Connection to an Oracle WebLogic Server**

1.2.1 Task 1: Install Oracle JDeveloper Studio

Install Oracle JDeveloper 11g Studio Edition to create the WebLogic Fusion Order Demo application. You can download Oracle JDeveloper from:


Ensure that you download and install 11g and that it is the Studio Edition, not the Java Edition. You can verify these details in Oracle JDeveloper from the Help > About menu option.

In order to create and deploy SOA composite applications and projects, you must install the Oracle SOA Suite extension. For instructions on installing this extension for Oracle JDeveloper, see the Oracle Fusion Middleware Installation Guide for Oracle JDeveloper.

1.2.2 Task 2: Install the Fusion Order Demo Application

 Throughout this tutorial, you must view or use content from Fusion Order Demo in your Oracle JDeveloper environment. The Fusion Order Demo is contained within a ZIP file.

To access the ZIP file:

1. Download the Fusion Order Demo application ZIP file (FusionOrderDemo_R1.zip). You can download the ZIP file from:


2. Unzip the file to a temporary directory.

   This tutorial refers to this directory as DEMODOWNLOAD_HOME. When you create the WebLogic Fusion Order Demo application, create the application in a working application directory, such as C:\fod. This tutorial refers to the working application directory location as MY_FOD_HOME. When requested, copy needed files from the DEMODOWNLOAD_HOME directory to MY_FOD_HOME.

1.2.3 Task 3: Install Oracle SOA Suite

To successfully deploy and run the Fusion Order Demo applications, you must complete an installation for Oracle SOA Suite. Installing Oracle SOA Suite requires creating schemas for Oracle SOA Suite in an Oracle database, installing Oracle WebLogic Server, installing Oracle SOA Suite, and configuring a domain in Oracle WebLogic Server to support both Oracle SOA Suite and Oracle Enterprise Manager. Specifically, the domain contains an Administration Server and a Managed Server. The Administration Server hosts Oracle Enterprise Manager Fusion Middleware Control
for performing administrative tasks; the Managed Server is an instance of an Oracle WebLogic Server used to host deployed applications. For instructions on installing and configuring Oracle SOA Suite, see the Oracle Fusion Middleware Installation Guide for Oracle SOA Suite.

After successfully completing the installation process, perform the following additional configuration steps:

1. Enable the credentials that are included in the StoreFront module by adding a setting to the configuration file for the domain:
   
a. Locate the configuration file set for the Oracle SOA Suite domain in the following directory:
      (UNIX) $MW_HOME/user_projects/domains/domain_name/bin/setDomainEnv.sh
      (Windows) $MW_HOME\user_projects\domains\domain_name\bin\setDomainEnv.cmd
   
b. Add the following option to the JAVA_PROPERTIES (UNIX) or the SET JAVA_PROPERTIES (Windows) line:
      -Djps.app.credential.overwrite.allowed=true
   
c. If the Oracle WebLogic Server Administration Server is running, stop it:
      On UNIX, as the root user, change directories to directory $MW_HOME/user_projects/domains/domain_name/bin and enter the following command:
      ./stopWebLogic.sh
      
      On Windows, from the Windows Start menu, select All Programs > Oracle WebLogic > User Projects > domain_name > Stop Admin Server.
   
d. Start the Administration Server:
      On UNIX, from directory $MW_HOME/user_projects/domains/domain_name/bin, enter the following command:
      ./startWebLogic.sh
      
      On Windows, from the Windows Start menu, select All Programs > Oracle WebLogic > User Projects > domain_name > Start Admin Server.
      
      When prompted on UNIX, enter your Oracle WebLogic Server user name and password. The password is not visible as you type.
      
      The Administration Server is started when the command window displays the following messages:
      <Server state changed to RUNNING>
      <Server started in RUNNING mode>
      
      Leave the command window open, although you may minimize it. The Administration Server is now running and ready for use.
   
e. When the Administration Server is in RUNNING mode, start the SOA managed server, if it is not running. In a command window, enter the following command all on one line:
      On UNIX, from directory $MW_HOME/user_projects/domains/domain_name/bin, enter the following command:
      ./startManagedWebLogic.sh managed_server_name admin_url username password
On Windows, from directory `MW_HOME\user_projects\domains\domain_name\bin`, enter the following command:
```
startWebLogic.cmd managed_server_name admin_url username password
```

Substitute the following values:

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>managed_server</code></td>
<td>The name of the Managed Server. For example:</td>
</tr>
<tr>
<td></td>
<td><code>soa_server1</code></td>
</tr>
<tr>
<td><code>admin_url</code></td>
<td>The URL of the Oracle WebLogic Server. For example:</td>
</tr>
<tr>
<td></td>
<td><code>http://soahost:7001</code></td>
</tr>
<tr>
<td><code>username</code></td>
<td>The Oracle WebLogic Server administrator. For example:</td>
</tr>
<tr>
<td></td>
<td><code>weblogic</code></td>
</tr>
<tr>
<td><code>password</code></td>
<td>The password of the Oracle WebLogic Server administrator. For example:</td>
</tr>
<tr>
<td></td>
<td><code>welcome1</code></td>
</tr>
</tbody>
</table>

2. If you are deploying remotely from one computer that has Oracle JDeveloper to another computer that has the Oracle SOA Suite installation with Oracle WebLogic Server, modify the `JAVA_HOME` and `PATH` environment variables on the computer with the Oracle SOA Suite installation.

Oracle JDeveloper requires changes to these variables for running the scripts that deploy the composite services. You set the `JAVA_HOME` variable to include the path to the Oracle WebLogic Server JDK, and set the `PATH` variable to include the path to the Oracle WebLogic Server `bin` directory for `ant`.

On UNIX, use the `export` command. For example:
```
export JAVA_HOME=$MW_HOME/jdk160_11
export PATH=$PATH:$MW_HOME/modules/org.apache.ant_1.7.0/bin
```

On Windows, perform the following steps to modify the variables:

a. Open Control Panel from the Windows Start menu and double-click the `System` icon.

b. In the System Properties dialog, select the `Advanced` tab and click `Environment Variables`.

c. In the Environment Variables dialog, locate the `JAVA_HOME` system variable and ensure that it is set to the location of the Oracle WebLogic Server JDK.

If there is no `JAVA_HOME` variable defined, click `New` and in the New System Variable dialog, enter a variable name of `JAVA_HOME` and a variable value pointing to the Oracle WebLogic Server JDK, such as `C:\weblogic\jdk160_11`. Click `OK` to set the new system variable.

d. Double-click the `Path` system variable and ensure that it includes the path to the Oracle WebLogic Server `ant\bin` directory. If it does not, add the path to the end of the variable value. For example:
```
;C:\weblogic\modules\org.apache.ant_1.7.0\bin
```

Click `OK` to set the new system variable.
e. Click **OK** twice more to dismiss the Environment Variables and the System Properties dialogs.

### 1.2.4 Task 4: Create a Connection to an Oracle WebLogic Server

To create a connection to the Oracle WebLogic Server configured for Oracle SOA Suite during installation.

1. Start Oracle JDeveloper:
   - (UNIX) `ORACLE_HOME/jdev/bin/jdev`
   - (Windows) `JDEV_ORACLE_HOME/jdeveloper/JDev/bin/jdev.exe`

2. From the **Application Menu**, select **New**.

3. In the New Gallery dialog, in the **Categories** tree, select **General**, and then **Connections**.

4. Select **Application Server Connection** and click **OK**.

   The Create Application Server Connection Type page displays.

5. Enter **MyAppServerConnection** in the **Connection Name** field and select **WebLogic 10.3** from the **Connection Type** list.
6. Click Next.
   The Authentication page is displayed.

7. Enter weblogic for the User Name and the password for that administrator in the Password field.

8. In the Configuration page, enter the following values:

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weblogic Hostname (Administration Server)</td>
<td>Name of the DNS name or IP address of the Administration Server of the Oracle WebLogic Server</td>
</tr>
<tr>
<td>Port</td>
<td>The address of the port on which the Administration Server is listening for requests (7001 by default)</td>
</tr>
<tr>
<td>WLS Domain</td>
<td>The domain name for Oracle WebLogic Server</td>
</tr>
</tbody>
</table>

9. Click Next.
   The Test page displays.

10. Click Test Connection.
    The following message should appear:

    Testing JSR-88                         ... success.
    Testing JSR-88-LOCAL                   ... success.
    Testing JNDI                           ... success.
    Testing JSR-160 DomainRuntime          ... success.
    Testing JSR-160 Runtime                ... success.
    Testing JSR-160 Edit                   ... success.
    Testing HTTP                           ... success.
    Testing Server MBeans Model            ... success.

   8 of 8 tests successful.

   If the test is unsuccessful, ensure that Oracle WebLogic Server is running, and retry the test.

11. Click Finish.

12. In the Resource Palette, under IDE Connections, expand Application Server to see the application server connection that you created.

1.3 Viewing the WebLogicFusionOrderDemo Application Artifacts

This tutorial focuses on building the WebLogic Fusion Order Demo application for Fusion Order Demo. To begin, spend time viewing the WebLogicFusionOrderDemo application artifacts in Oracle JDeveloper:

1. Start Oracle JDeveloper.

2. From the JDeveloper main menu, choose File > Open.
3. In the Open dialog, browse to `DEMO_DOWNLOAD_HOME/CompositeServices` and select `WebLogicFusionOrderDemo.jws`. Click Open.

The following figure shows the Application Navigator after you open the file for the application workspace.

![Application Navigator](image)

The following figure shows the Application Navigator after you open the file for the application workspace.

Table 1–1 describes each of the projects in the `WebLogicFusionOrderDemo` application workspace:

<table>
<thead>
<tr>
<th>Project</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>B2BX12OrderGateway</td>
<td>This project contains a composite for Oracle B2B. This composite is not used in this tutorial.</td>
</tr>
<tr>
<td>BamOrderBookingComposite</td>
<td>This project contains the OrderBookingComposite composite with Oracle BAM addition. Specifically, it uses the Oracle BAM adapter and Oracle BAM sensors to send active data into Oracle BAM dashboard. This composite is not used in this tutorial.</td>
</tr>
<tr>
<td>bin</td>
<td>This project contains a build script for deploying all the SOA projects. It also contains templates for seeding JMS connector information, demo topics, and demo users.</td>
</tr>
<tr>
<td>CreditCardAuthorization</td>
<td>This project provides the service needed by OrderBookingComposite project to verify the credit card information of a customer.</td>
</tr>
<tr>
<td>OrderApprovalHumanTask</td>
<td>This project provides a task form for approving orders from the OrderBookingComposite project.</td>
</tr>
<tr>
<td>OrderBookingComposite</td>
<td>This project processes an order submitted in the Store Front module UI. This project contains the main process for the WebLogic Fusion Order Demo application.</td>
</tr>
<tr>
<td>OrderSDComposite</td>
<td>This project simulates the StoreFrontService service of the Store Front module for testing purposes.</td>
</tr>
<tr>
<td>PartnerSupplierComposite</td>
<td>This project contains a composite containing a BPEL process for obtaining a quote from a partner warehouse. It is referenced as a service from the composite for the OrderBookingComposite project.</td>
</tr>
</tbody>
</table>

4. From the Application Menu, select Close to close the sample application.

### 1.4 Understanding the OrderBookingComposite Flow

Composites enable you to easily assemble multiple technology components into one SOA application. A composite groups service components and uses wires to connect components. `OrderBookingComposite` is the main project of the WebLogic Fusion
Order Demo application, containing a composite application for processing orders from Global Company. This composite demonstrates how services, both internal to an enterprise, and external at other sites, can be integrated using the SOA architecture paradigm to create one cohesive ordering system.

The OrderBookingComposite composite utilizes the following Oracle SOA Suite components:

- Oracle Mediator
- Oracle BPEL Process Manager
- Oracle Human Workflow (using a human task)
- Oracle Business Rules
- Oracle Messaging Service

At the center of OrderBookingComposite composite is the OrderProcessor BPEL process. It orchestrates all the existing services in the enterprise for order fulfillment with the right warehouse, based on the business rules in the process.

Figure 1–1 shows an overview of the OrderBookingComposite composite for the WebLogic Fusion Order Demo application, followed by a step-by-step description of the composite flow for how the application processes an order.
When a new customer registers in Global Company’s storefront UI, the Web client sends the customer’s information to the internal customer service application called StoreFrontService. StoreFrontService then stores the customer information in a database. The customer can then browse products, add them to their online shopping cart, and place the order.

When a registered customer attempts onto Global Company’s storefront UI, the UI invokes the StoreFrontService and provides authentication. A registered user
builds up their shopping cart, and places an order. When the order is submitted, the following events take place:

After an order is placed, the following sequence occurs to complete the order:

1. Oracle ADF Business Component writes the order to a database with schema for Fusion Order Demo, and raises a NewOrderSubmitted event using the Event Delivery Network (EDN). The data associated with this event is the order ID.

2. Because the OrderPendingEvent mediator subscribes to the NewOrderSubmitted event, the EDN layer notifies the OrderPendingEvent mediator of the new order.

3. The OrderPendingEvent mediator receives the order and routes the input order ID to the OrderProcessor BPEL process.

4. The OrderProcessor BPEL process receives the order ID from the database, using a bind entity activity to bind to the exposed Oracle ADF Business Component StoreFrontService service.

Some of the information about the order used later in the process is:

- Customer ID
- Items the customer purchased
- Credit card used
- Shipping address chosen

5. The BPEL process initiates StoreFrontService, passing it the order ID, to retrieve information about the customer.

6. The BPEL process then sends the purchase amount, credit card type, and credit card number to CreditCardAuthorizationService, which verifies if the customer’s credit card is valid.

   If credit card is not valid, the BPEL process cancels the order.

   If credit card is valid, the BPEL process sends the order to the RequiresApprovalRule business rule to determine if the order requires approval by management.

7. The RequiresApprovalRule business rule evaluates if manual approval is required. The business rule contains a rule that requires manual approval for orders over $2,000.

8. For those orders requiring manual approval, the BPEL process invokes the ApprovalHumanTask human task, which routes a message to a manager, who then approves or disapproves the order.

9. If the order is approved, the BPEL process sends the order information to the following suppliers in parallel to obtain a bid:

   - Internal supplier by using the InternalWarehouseService BPEL process, also located in OrderBookingComposite

   - External supplier by using the PartnerSupplierMediator mediator, which in turn routes to the ExternalPartnerSupplier BPEL process, located in another composite called PartnerSupplierComposite

10. The two suppliers respond with their bids, and the BPEL process send the bids to the EvaluatePreferredSupplierRule business rule.

11. The EvaluatePreferredSupplierRule business rule chooses the supplier with the lower of the two bids.
12. The BPEL process invokes the FulfillOrder mediator, which performs the following two operations:
   - Stores the order in a temporary queue and uploads it to the fulfillment system in batch mode overnight
   - Routes the order to USPS
13. Once the order is fulfilled, the BPEL process sets the order to complete.
14. The BPEL process invokes the NotificationService service, which sends the customer an E-mail notification with the purchase order information.
15. When the order completes, the OrderPendingEvent mediator publishes the OrderCompleted business for the OrderProcessor process.

When an order is updated, the following occurs:

1. The UpdateOrderStatus mediator publishes business event OrderUpdateEvent and sends the order ID to the OrderProcessor BPEL process.
2. The OrderUpdateEventMediator mediator subscribes to business event OrderUpdateEvent, sends the order ID to StoreFrontService, and waits for the StoreFrontService to respond with updated details about the order.
This chapter describes how to deploy Fusion Order Demo, place an order, and monitor the order as it is processed by the WebLogic Fusion Order Demo application. It explains two different order scenarios for the Web client and how to monitor orders processed through the business flow.

Before following the instructions in this chapter, perform all the procedures in Chapter 1.

This chapter includes the following sections:

- Section 2.1, "Deploying the Fusion Order Demo Applications"
- Section 2.2, "Placing Two Orders in the Store Front"
- Section 2.3, "Starting Fusion Middleware Control to Monitor Orders"
- Section 2.4, "Monitoring the First Order"
- Section 2.5, "Monitoring the Second Order"
- Section 2.6, "Undeploying the Composites for the WebLogic Fusion Order Demo Application"

2.1 Deploying the Fusion Order Demo Applications

To run the demo, deploy the applications for the Store Front module and the WebLogic Fusion Order Demo application, performing the following tasks:

- Task 1: Install the Schema for the Fusion Order Demo Application
- Task 2: Deploy the Store Front Module
- Task 3: Deploy the WebLogic Fusion Order Demo Application

2.1.1 Task 1: Install the Schema for the Fusion Order Demo Application

To install the schema for the sample application:

1. Start Oracle JDeveloper 11g and from the main menu choose File > Open.
2. In the Open dialog, browse to DEMO_DOWNLOAD_HOME/Infrastructure and select Infrastructure.jws. Click Open.
3. When prompted to migrate files to the 11.1.1.0 format, click Yes. When the migration is complete, click OK.
4. In the Application Navigator, expand MasterBuildScript and then Resources, and double-click build.properties.
5. In the editor, modify the following properties for your environment:

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>jdeveloper.home</td>
<td>The root directory where you have Oracle JDeveloper 11g installed. For example: C:/JDeveloper/11</td>
</tr>
<tr>
<td>jdbc.urlBase</td>
<td>The base JDBC URL for your database in the format jdbc:oracle:thin:@&lt;yourhostname&gt;. For example: jdbc:oracle:thin:@foddb-server</td>
</tr>
<tr>
<td>jdbc.port</td>
<td>The port for your database. For example: 1521</td>
</tr>
<tr>
<td>jdbc.sid</td>
<td>The SID of your database. For example: ORCL or XE</td>
</tr>
<tr>
<td>db.adminUser</td>
<td>The administrative user for your database. For example: system</td>
</tr>
<tr>
<td>db.demoUser.tablespace</td>
<td>The tablespace name for the Fusion Order Demo users. For example: USERS</td>
</tr>
</tbody>
</table>

6. From the JDeveloper main menu, choose **File > Save All**.

7. In the Application Navigator, under the **Resources** node, right-click **build.xml** and choose **Run Ant Target > buildAll**.

8. When prompted, enter the administrative-user password for your database.

   The **buildAll** command then creates the **FOD** user and populates the tables in the **FOD** schema. In the Apache Ant - Log, a series of SQL scripts display, followed by:

   buildAll: BUILD SUCCESSFUL
   Total time: nn minutes nn seconds

   For more information on the demo schema and scripts, see the **README.txt** file in the **MasterBuildScript** project.

### 2.1.2 Task 2: Deploy the Store Front Module

You place orders by running the **home.jspx** page in the **StoreFrontUI** project of the Store Front module. The **StoreFrontUI** project uses JavaServer Faces (JSF) as the view technology, and relies on the Oracle ADF Model layer to interact with Oracle ADF Business Components in the **StoreFrontService** project.
Deploying the Fusion Order Demo Applications

Running the Sample Application 2-3

Figure 2–1  StoreFrontUI Home Page

From the home page, you can browse the Web site as an anonymous user, then log in as a registered customer to place an order.

The Fusion Order Demo application ships with predefined customer data. Because the Fusion Order Demo application implements Oracle ADF Security to manage access to Oracle ADF resources, only the authenticated user can view orders in their cart. The following table shows the preregistered customers. You place orders as ngreenbe later in this chapter.

<table>
<thead>
<tr>
<th>Username</th>
<th>Password</th>
<th>Application Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>ngreenbe</td>
<td>welcome1</td>
<td>Customer (CUST)</td>
</tr>
<tr>
<td>skying</td>
<td>welcome1</td>
<td>Staff (STAFF)</td>
</tr>
<tr>
<td>pbrown</td>
<td>welcome1</td>
<td>Supplier (SUPP)</td>
</tr>
</tbody>
</table>

To learn more about the Store Front module and to understand its implementation details, see Oracle Fusion Middleware Fusion Developer’s Guide for Oracle Application Development Framework.

To deploy the Store Front module:

1. Choose File > Open.
2. In the Open dialog, browse to DEMO_DOWNLOAD_HOME/StoreFrontModule and select StoreFrontModule.jws. Click Open.
3. When prompted to migrate files to the 11.1.1.0 format, click Yes. When the migration is complete, click OK.

The following figure shows the Application Navigator after you open the file for the application workspace.
4. Deploy the services used by the Store Front module to send orders to the
OrderBookingComposite composite.
   a. In the Application Navigator, right click StoreFrontModule and choose
Deploy > StoreFrontModule_SDOServices > to > MyAppServerConnection.
   You created this connection in Section 1.2.4, "Task 4: Create a Connection to an
   Oracle WebLogic Server."
   b. In the Select Deployment Targets dialog, select the Managed Server for the
   Oracle WebLogic Server, such as soa_server, and click OK.
   c. In the Deployment Configuration dialog, accept the default MDS repository
   name and partition name, and then click Deploy.

5. Deploy the Store Front module. From the Application menu, select Deploy >
StoreFrontModule > to > MyAppServerConnection.

2.1.3 Task 3: Deploy the WebLogic Fusion Order Demo Application

To deploy the WebLogic Fusion Order Demo application to an Oracle SOA Suite
installation, containing an Oracle WebLogic Server domain with an Administration
Server and a Managed Server.

1. In the Application Navigator, select WebLogicFusionOrderDemo.
2. Expand bin and then Resources, and double-click build.properties.
3. In the editor, modify the following properties for your environment:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>oracle.home</td>
<td>The root directory where you have Oracle JDeveloper 11g installed. For example: C:\Oracle\Middleware\jdeveloper\</td>
</tr>
<tr>
<td>soa.only.deployment</td>
<td>false</td>
</tr>
<tr>
<td>admin.server.host</td>
<td>The DNS name or IP address of the Administration Server for Oracle SOA Suite for hosting applications. For example: soahost</td>
</tr>
<tr>
<td>admin.server.port</td>
<td>The port of the Administration Server. For example: 7001</td>
</tr>
<tr>
<td>managed.server</td>
<td>The DNS name or IP address of the Managed Server for Oracle SOA Suite for hosting applications. For example: soahost</td>
</tr>
</tbody>
</table>
From the JDeveloper main menu, choose **File > Save All**.

In the Application Navigator, under the **Resources** node, right-click **build.xml** and choose **Run Ant Target** and select the following ant targets in the following sequential order:

<table>
<thead>
<tr>
<th>Target</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. setupWorkspaceForJDeveloperUse</td>
<td>This script sets up the application workspace in Oracle JDeveloper.</td>
</tr>
<tr>
<td>2. seedPodJmsResources</td>
<td>This script populates the JMS resources for the Fulfillment mediator.</td>
</tr>
<tr>
<td>3. seedDemoUsers</td>
<td>This script adds jstein as the user to approve orders for over $2,000. When you run the demo, you place an order for $2,000 and log in to the Oracle BPM Worklist as jstein and approve the order. If you do not run this script, the OrderProcessor BPEL process generates a recoverable error and assigns the task to the weblogic administrator.</td>
</tr>
<tr>
<td>4. compile-build-all</td>
<td>This script compiles and builds all the SOA composites.</td>
</tr>
</tbody>
</table>
Do not run the next target in the sequence until the previous one completes successfully. In the Apache Ant - Log, ensure you see the following message before proceeding to the next script:

BUILD SUCCESSFUL
Total time: nn minutes nn seconds

For more information about the ant targets, see the following resources:

- Appendix B, "ant Scripts"
- Readme.txt file in the DEMO_DOWNLOAD_HOME/CompositeServices/

### 2.2 Placing Two Orders in the Store Front

The ordering process begins in the storefront UI, where a user shops for and orders products. When an order is submitted, the Application Development Framework Business Component writes the order to database and raises an NewOrderSubmitted business event using the Events Delivery Network (EDN). The OrderPendingEvent mediator subscribes this event, and initiates the main BPEL process, OrderProcessor, to process the order.

In this task, place two orders, one for under $2000 and the other for over $2000. By placing these order, you can see how orders totalling more than $2000 require human approval.

To place orders:

1. Access the storefront from the following URL:

   http://hostname:port/StoreFrontModule/faces/home.jspx

   where hostname is the DNS name or IP address of the Oracle WebLogic Server for Oracle SOA Suite and port is the address of the port on which the Managed Server for the Oracle WebLogic Server is listening for requests (8001 by default).

   You begin the order process by browsing the product catalog. When you click Add next to a product, the site updates the shopping cart region to display the item.

2. Place an order for under $2000:

   a. Click the Ipod Nano 1 Gb for $149.95 and click Add.

   b. Click the Ipod Nano 2 Gb for $199.95 nd click Add.

   The following shows the cart summary with the two products added. The summary displays a subtotal purchase total of $349.90 for the two items that appear in the cart.
c. Click the **Checkout** link, located in the middle of the menu bar. A login dialog displays.

d. Enter `ngreenbe` and `welcome` in the **Username** and **Password** fields, respectively.

e. Click **Log In**.

The Shipping Details page displays.
f. In the Order Information #order_id section, take note of the order ID, as you need it for a later task. In this example, the order ID is 1179.

g. In the Shipping Information section, take note of the following default settings:
   - 100 N Peach St Philadelphia PA 19139 US
   - 536267 MSTR

h. In the Order Summary section, take note of the order total, which is $353.86 and click Submit Order to complete the first order. The Invoice Details page displays.

3. Click the Exit and Continue Shopping link, located in the menu bar, to return to the home page.
4. Place an order for over $2000:
   a. Click the Plasma HD Television for $1,999.99 and click Add.
   b. Click the Playstation 2 Video Game for $199.95 and click Add.
      The Shopping Cart Summary section displays a subtotal purchase total of $2,199.94 for the two items that appear in the cart.
   c. Click the Checkout link.
      The Shipping Details page displays.
   d. In the Order Information #order_id section, take note of the order ID, as you need it for a later task. In this example, the order ID is 1180.
   e. In the Shipping Information section, modify any of the options, such as the address in the Shipping Address field and the credit card in the Payment Options section. Take note of the settings you select.
      - 2100 S Casino Dr. Laughlin NV 89029 US
      - 4111111111111111 VISA
   f. In the Order Summary section, take note of the order total, which is $2,219.42 and click Submit Order to complete the second order.
      The Invoice Details page displays.

5. Click the Logout link, located in the right-hand-side of the menu bar, to log out of the session.
2.3 Starting Fusion Middleware Control to Monitor Orders

To start Oracle Enterprise Manager Fusion Middleware Control to monitor orders:

1. Use Mozilla FireFox, version 2.0 or higher, or Internet Explorer, version 7.0 or higher, to access the following URL:
   
   http://hostname:port/em

   where hostname is the host name and port is the port of the Fusion Middleware Control for the Oracle SOA Suite installation.

   The login dialog appears.

2. Enter weblogic/password and click Login

   where:
   
   - weblogic is the Fusion Middleware Control Console administrator user name
   - password is the password you provided for the weblogic administrator during installation

3. From the navigation pane, expand SOA > soa-infra to see the deployed applications:

   OrderBookingComposite and PartnerSupplierComposite are the two main composites in the WebLogic Fusion Order Demo application.

2.4 Monitoring the First Order

You now monitor the first order sent from the Store Front module to the OrderBookingComposite composite from the Fusion Middleware Control. This order was submitted by Nancy Greenberg for $353.86.

To monitor the first order:

1. From the navigation pane, expand SOA > soa-infra to see the deployed applications:
OrderBookingComposite and PartnerSupplierComposite are the two main composites in the WebLogic Fusion Order Demo application.

2. Click the OrderBookingComposite composite.

The SOA Infrastructure home page displays. The upper part of the page displays details about recent SOA composite application instances, recent faults, and rejected messages. The topmost two instances in the Recent Instances table represents the instance of the SOA composite application you created when you placed orders in Section 2.2, "Placing Two Orders in the Store Front."

3. In the Recent Instances section, click the second instance, representing the order for $353.86. Note the instance number.

The Flow Trace page displays. The Trace section shows the sequence of the message flow through the services, service components, and references that comprise the SOA composite application.
The Trace section displays the state of each service component within the OrderBookingComposite composite. Notice all the service components have a status of Completed, indicating that the order was successfully processed.

4. Click the OrderProcessor BPEL process service component in the Instance column to view instance more carefully.

The Audit Trail tab of the Instance of OrderProcessor window displays execution details about the activities in the BPEL process. Notice, too, the instance you selected displays in the Instance ID field on the right side.

5. Click the Flow tab to see a visual representation of the instance.

A visual representation of the BPEL process activities appears. The icons in the flow are referred to as activities. You can click them to view their details.
Click the first activity, the blue, circle receive activity labeled **receiveInput**.

The Activity Details dialog displays the XML input to this BPEL process instance. It shows the order ID (**orderID**) you were given when you placed the order through the Store Front module. In this example, the order ID is 1179.

7. Click **X** or **Close** to dismiss the Activity Details dialog.

8. Scroll further in the process to the **Scope_RetrieveOrder** activity and click **findOrderByID** to order ID as input to variable **gOrderInfoVariable**:

   ```
   bpelx:bindEntity is executed on variable gOrderInfoVariable with the following key values:
   
   {{/oracle/fodemo/storefront/store/queries/common/}OrderId=1179}
   ```
9. Click X or Close to dismiss the Activity Details dialog.

10. Scroll to the Scope_RetrieveCustomerForOrder scope and click the InvokeFindCustomer activity to see the StoreFrontService service being invoked and returning the customer information. Note the following in the Activity Details window:

- lFindCustomerInfo_InputVariable represents the input variable to the StoreFrontService service. The CustId parameter represents the ID of the customer of customer ngreenbe.
- gCustomerInfoVariable represents the output from the StoreFrontService service, which returns the customer information back to the BPEL process.

11. Click X or Close to dismiss the Activity Details dialog.

12. Scroll to the Scope_AuthorizeCreditCard scope and click InvokeCheckCreditCard to see the CreditCardAuthorization service being invoked and returning the status of the credit card.

Note the following in the Activity Details window:

- lCreditInput represents the input variable to the CreditCardAuthorizationService service. Parameters CCType, CCNumber, and PurchaseAmount represent the input credit card type, number, and order total. These values are set as follows:

```xml
<CCType>MSTR</CCType>
<CCNumber>536267</CCNumber>
<PurchaseAmount>353.86</PurchaseAmount>
```

- lCreditCardOutput shows the service returned a status of APPROVED. Therefore, the OrderProcessor BPEL process continues.

13. Click X or Close to dismiss the Activity Details dialog.
14. Scroll to the **Scope_CheckApprovalLimit** scope and click the **Invoke** link under the **BusinessRule_ApprovalRequired** to see the **RequiresApprovalRule** business rule being invoked.

Note the following in the Activity Details window:

- **dsIn** represents the input variable sent to the business rule. The **approve** parameter shows the invocation of the rules engine at runtime. The **price** parameter shows the order total was $353.86. The rule engine requires this input to determine whether human approval is required.

- **dsOut** shows the output from the business rule. The **approvalRequired** parameter has a value **false**. Because this order is under $2,000, no human approval is required. Therefore, the return value is set to **false**.

15. Click **X** or **Close** to dismiss the Activity Details dialog.

16. In the **Scope_RetrieveQuotes** scope, perform the following:

   a. Click the **Invoke_PartnerSupplier** link to see the order information being sent to the **PartnerSupplierMediator** mediator. The **PartnerSupplierMediator** routes the order information to the **ExternalPartnerSupplier** BPEL process, located in **PartnerSupplierComposite** composite.
b. Click X or Close to dismiss the Activity Details dialog.

c. Click the **Invoke_InternalWarehouse** link to see the order ID being sent to the **InternalWarehouseService** BPEL process through the **lInternalWarehouseInputVariable** variable:

![Invoke_InternalWarehouse](image)


d. Click X or Close to dismiss the Activity Details dialog.

17. In the **Scope_SelectPreferredSupplier** scope, click the **Invoke** link under **BusinessRule_SelectPreferredSupplier** to see the **EvaluatePreferredSupplierRule** business rule being invoked.

Note the following in the Activity Details window:

- **dsIn** represents the input variable sent to the business rule. The **warehouse**, **deliveryDate**, and **orderTotal** parameter values provide the input to the business rule. The rule engine uses this input to pick the supplier with the lowest shipping price to fulfill the order.

The returned input data for the two warehouse suppliers is as follows:

```xml
<warehouse>InternalWarehouse</warehouse>
<deliveryDate>2009-02-13</deliveryDate>
<orderTotal>1000</orderTotal>
...
<warehouse>PartnerWarehouse</warehouse>
<deliveryDate>2009-02-13</deliveryDate>
```
The InternalWarehouse supplier returns a static value of $1,000 for all orders.

- dsOut shows the output from the business rule. The warehouse parameter value shows the selected warehouse supplier. The PartnerWarehouse supplier was selected, because it provided a lower quote.

18. Click X or Close to dismiss the Activity Details dialog.

19. In the Scope_FulfillOrder scope, click Invoke_FulfillOrder link to see the order information being sent to the FulfillOrder mediator. The FulfillOrder mediator stores the order in a temporary queue and routes the order to USPS for shipment.

20. Click X or Close to dismiss the Activity Details dialog.

21. Scroll to the Scope_NotifyCustomerOfCompletion scope and click the InvokeNotificationService link to see the output E-mail notification sent to Nancy Greenberg.

22. Click X or Close to dismiss the Activity Details dialog.

23. Close the Flow Trace window.
2.5 Monitoring the Second Order

You now monitor the second order you submitted for $2,219.42 as Nancy Greenberg. This order is processed differently than the first order, because the order amount is for more than $2,000. Therefore, it requires human approval. In this task, monitor the order with the Fusion Middleware Control, approve the order with Oracle BPM Worklist, and see the order complete in the Fusion Middleware Control.

To monitor the second order, complete the following tasks:

- Task 1: View the Order in the Fusion Middleware Control
- Task 2: Use the Oracle BPM Worklist to Approve the Order
- Task 3: View the Approval in the Fusion Middleware Control

2.5.1 Task 1: View the Order in the Fusion Middleware Control

1. From the SOA Infrastructure home page, in the Recent Instances section, click the first instance, representing the order for $2,219.42.
   
The Flow Trace page displays.

2. Click the OrderProcessor BPEL process service component in the Instance column.
   
The Audit Trail tab of the Flow Trace window displays execution details about the activities in the BPEL process.

3. In the Recent Instances section, click the first instance, representing the order for 2,219.42. Note the instance number.
   
The Flow Trace page displays.

   ![Image of Flow Trace page](image)

   Unlike the first order, notice in the Trace section how the service components in the OrderBookingComposite composite are not all complete and the process is stopped at the ApprovalHumanTask component.

4. Click the OrderProcessor BPEL process service component in the Instance column to look at the instance more carefully in the Instance of OrderProcessor window.

5. Click the Flow tab to see a visual representation of the instance.

6. Click the receiveInput activity to see the order ID you placed.
The Activity Details dialog displays the XML input to this BPEL process instance. It shows the order ID (orderID) you were given when you placed the order through the Store Front module. In this example, the order ID is 1180.

7. Click X or Close to dismiss the Activity Details dialog.

8. Scroll to the Scope_CheckApprovalLimit scope and click the Invoke link under the BusinessRule_ApprovalRequired to see the RequiresApprovalRule business rule being invoked.

Note the following in the Activity Details window:

- dsIn represents the input variable sent to the business rule. The approve parameter shows the invocation of the rules engine at run time. The price parameter shows the order total was $2,219.42.
- dsOut shows the output from the business rule. The approvalRequired parameter has a value true. Because this order is over $2,000, human approval is required. Therefore, the return value is set to true.

9. Click the Collapse icon (-) icon on the BusinessRule_ApprovalRequired to collapse it.

10. Scroll toward the bottom of the Scope_CheckApprovalLimit scope to see the ApprovalHumanTask human task was initiated after the business rule. The switch under the business rule is like a if-then-else, case, or switch
statement in other programming languages. In this case, if approval is required, this switch submits data to the ApprovalHumanTask human task.

11. Click the initiateTask_ApprovalHumanTask_1 activity.

12. In the Activity Details dialog, scroll to the initiateTaskResponseMessages variable to see the assigneeUsers parameter. This parameter specifies the human task has been assigned to user jstein:

   - <assigneeUsers>
     <id>jstein</id>
     <type>user</type>
   </assigneeUsers>

13. Click X or Close to dismiss the Activity Details dialog.

14. Click the receiveCompletedTask_ApprovalHumanTask_1 activity to see the human task is awaiting approval.

15. Click X or Close to dismiss the Activity Details dialog, but do not close the Flow Trace page, as you need it to view order processing when it completes.
2.5.2 Task 2: Use the Oracle BPM Worklist to Approve the Order

To approve the order, use the Oracle BPM Worklist:

1. Use Internet Explorer 7 or Mozilla Firefox 2.0.0.2 to access the following URL:
   
   \[
   \text{http://hostname:port/integration/worklistapp/faces/home.jspx}
   \]
   
   where `hostname` is the DNS name or IP address of the Oracle WebLogic Server for Oracle SOA Suite and `port` is the address of the port on which the Managed Server for the Oracle WebLogic Server is listening for requests (8001 by default). The login dialog appears.

2. Enter `jstein` in the **Username** field and `welcome1` in the **Password** field and click **Login**.

3. The **Inbox** shows the order number 1180 is awaiting approval.

   ![BPM Worklist Image]

   4. Select the order from the table and from the **Actions** menu, select **Approve**.

   ![Actions Menu Image]

   5. Click **OK** to acknowledge the approval message.

      The **My Tasks** tab updates so that no worklist tasks are currently assigned to `jstein`.

      With the order approval completed, the processing for the order is now complete.

2.5.3 Task 3: View the Approval in the Fusion Middleware Control

To verify the order completed processing with Fusion Middleware Control:

1. Go back to the Flow Trace page and click the **Refresh** icon, located in the top right corner:
2. Scroll toward the bottom of the **Scope_CheckApprovalLimit** scope to see the **ApprovalHumanTask** human proceeded.

3. Click the **receiveCompletedTask_ApprovalHumanTask_1** activity to see the human task.

4. Click X or **Close** to dismiss the Activity Details dialog.

5. In the **Scope_SelectPreferredSupplier** scope, click the **Invoke** link under **BusinessRule_SelectPreferredSupplier** to see the **EvaluatePreferredSupplierRule** business rule being invoked.

   Note the following in the Activity Details window:

   - **dsIn** represents the input variable sent to the business rule. The **warehouse**, **deliveryDate**, and **orderTotal** parameter values provide the input to the business rule. The rule engine uses this input to pick the supplier with the lowest shipping price to fulfill the order.

   The returned input data for the two warehouse suppliers is as follows:

   ```xml
   <warehouse>InternalWarehouse</warehouse>
   <deliveryDate>2009-03-13</deliveryDate>
   <orderTotal>1000</orderTotal>
   ...
   <warehouse>PartnerWarehouse</warehouse>
   <deliveryDate>2009-03-13</deliveryDate>
   <orderTotal>2219.42</orderTotal>
   ```
The warehouse parameter value shows the selected warehouse supplier. The InternalWarehouse supplier was selected, because it provided a lower quote.

6. Click X or Close to dismiss the Activity Details dialog.

7. In the Instance of OrderProcessor window, click the Flow Trace breadcrumb to return to the main Flow Trace window.

Notice in the Trace section how the service components in the OrderBookingComposite composite are now all complete.


2.6 Undeploying the Composites for the WebLogic Fusion Order Demo Application

The remainder of this tutorial describes how to build the composite applications for the WebLogic Fusion Order Demo applications. Because you deploy the composite applications during the development process, you can now undeploy the completed ones.

To undeploy the composite applications:

1. Access Undeploy SOA Composite wizard in Fusion Middleware Control through the following options:
The Confirmation page appears.

2. Click Undeploy. Note that you are warned if you are about to undeploy the last remaining revision of a deployed composite application.

Processing messages are displayed.

3. When undeployment has completed, click Close.
Creating the SOA Application

This chapter describes how to create the WebLogic Fusion Order Demo application in Oracle JDeveloper for Fusion Order Demo. It also describes how to create the PartnerSupplierComposite during the creation of the application.

Before following the instructions in this chapter, perform all the procedures in Chapter 1.

This chapter contains the following sections

- Section 3.1, "About the PartnerSupplierComposite Composite"
- Section 3.2, "Creating the WebLogicFusionOrderDemo Application and the PartnerSupplierComposite Composite"

See Chapter 1, "Introduction to the SOA Sample Application," for an overview of the WebLogic Fusion Order Demo application.

3.1 About the PartnerSupplierComposite Composite

The PartnerSupplierComposite composite contains a BPEL process, ExternalPartnerSupplier. In the OrderBookingComposite composite, the Scope_RetrieveQuotes scope of the OrderProcessor BPEL process uses this process to obtain a quote from an external partner warehouse. When you create the Scope_RetrieveQuotes scope and the ExternalPartnerSupplier Web service in Chapter 6, "Creating the Second Half of the OrderProcessor BPEL Process," you reference the ExternalPartnerSupplier BPEL process.

Figure 3–1 shows PartnerSupplierComposite in the SOA Composite Editor.
3.2 Creating the WebLogicFusionOrderDemo Application and the PartnerSupplierComposite Composite

In this procedure, you create the WebLogic Fusion Order Demo application, the PartnerSupplierComposite project, and the ExternalPartnerSupplier BPEL process. It contains the following tasks:

- Task 1: Create the WebLogicFusionOrderDemo Application and the PartnerSupplierComposite Composite
- Task 2: Create the ExternalPartnerSupplier BPEL Process
- Task 3: Modify the ExternalPartnerSupplier BPEL Process
- Task 4: Deploy the PartnerSupplierComposite Composite
- Task 6: Initiate a Test Instance for the PartnerSupplierComposite Composite

3.2.1 Task 1: Create the WebLogicFusionOrderDemo Application and the PartnerSupplierComposite Composite

To create the WebLogic Fusion Order Demo application:

1. Open the New Gallery by choosing File > New.
2. Click the Current Project Technologies tab.
3. From either the All Technologies tab or the Current Project Technologies tab, in the Categories tree, select General, and then Applications.
4. From the Items list, select SOA Application.
5. Click OK.

6. On the Name your application page, enter the following values:

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application Name</td>
<td>WebLogicFusionOrderDemo</td>
</tr>
<tr>
<td>Directory</td>
<td>Specify directory location directory_ path\CompositeServices, such as C:\fod\CompositeServices. Oracle JDeveloper creates this directory, which acts as a container for all the projects. This tutorial refers to the application directory that you specified as MY_FOD_HOME.</td>
</tr>
<tr>
<td>Application Package Prefix</td>
<td>Do not enter a value.</td>
</tr>
</tbody>
</table>

7. Click Next.

8. On the Name your project page, enter the following values:

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Name</td>
<td>PartnerSupplierComposite</td>
</tr>
<tr>
<td>Directory</td>
<td>Accept the default directory location, MY_FOD_HOME\CompositeServices\PartnerSupplierComposite. Oracle JDeveloper creates this directory for all the contents of the PartnerSupplierComposite composite project.</td>
</tr>
<tr>
<td>Project Technologies SOA</td>
<td>SOA</td>
</tr>
</tbody>
</table>

9. Click Next.

10. On the Configure SOA Settings page, from the Composite Template section, select Composite With BPEL.

11. Click Finish.

   The Create BPEL Process dialog displays.

### 3.2.2 Task 2: Create the ExternalPartnerSupplier BPEL Process

Now, continue with the creation of the ExternalPartnerSupplier BPEL process:

1. In the Create BPEL Process dialog, enter the following values:

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>ExternalPartnerSupplier</td>
</tr>
<tr>
<td>Namespace</td>
<td><a href="http://www.partnersupplier.example.com/ns/warehouse">http://www.partnersupplier.example.com/ns/warehouse</a></td>
</tr>
<tr>
<td>Template</td>
<td>Asynchronous BPEL Process</td>
</tr>
<tr>
<td>Expose as a SOAP service</td>
<td>Select this check box to create a BPEL process connected to an inbound SOAP service binding component.</td>
</tr>
</tbody>
</table>

2. In the Input field, import the complete schema located in the DEMO_DOWNLOAD_HOME directory.
   a. In the Input field, click the Browse Input Elements icon.
Creating the WebLogicFusionOrderDemo Application and the PartnerSupplierComposite Composite

The Type Chooser dialog displays.

b. Click the **Import Schema File** icon.

The Import Schema File dialog displays.

c. Click the **Browse Resources** icon to the right of the **URL** field.

The SOA Resource Browser displays.

d. Select **File System** and in the **Location** section, search for `Warehouse.xsd` in 
   `DEMO_DOWNLOAD_HOME/CompositeServices/PartnerSupplierComposite/xsd` and click **OK**.

e. In the Import Schema dialog, ensure the `Warehouse.xsd` now displays in the **URL** field and the **Copy to Project** option is selected, and then click **OK**.

   The Localized Files dialog displays, prompting you to import the `Warehouse.xsd` schema file and any dependent files, which includes the `ExternalPartnerSupplier.wsdl` WSDL file.

f. Deselect option **Maintain original directory structure for imported files** and click **OK** to import the files.

   The Type Chooser dialog displays.

g. Expand **Project Schema Files** > `Warehouse.xsd` and select `WarehouseProcessRequest` and then click **OK**.

The Type Chooser dialog displays.

3. In the Create BPEL Process dialog, import the elements from the `Warehouse.xsd` file for the output:

   a. In the **Output** field, click the **Browse Output Elements** icon.

   The Type Chooser dialog displays.
Creating the WebLogicFusionOrderDemo Application and the PartnerSupplierComposite Composite

b. Expand Project Schema Files > Warehouse.xsd and select WarehouseProcessResponse, and then click OK.

4. In the Create BPEL Process dialog, click OK.

The designer displays three tabs:

<table>
<thead>
<tr>
<th>Tab</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WebLogicFusionOrderDemo.jws</td>
<td>This tab shows the contents of the application workspace. The Application Overview is the home for all files you can create in this application. If this tab does not display, from the Application menu, select Show Overview. For an introduction to the Application Overview, click F1 on the tab to display the online help.</td>
</tr>
<tr>
<td>composite.xml</td>
<td>This tab displays the PartnerSupplierComposite composite in the SOA Composite Editor. For an overview of the SOA Composite Editor, see Oracle Fusion Middleware Developer’s Guide for Oracle SOA Suite.</td>
</tr>
<tr>
<td>ExternalPartnerSupplier.bpel</td>
<td>This tab displays the ExternalPartnerSupplier BPEL process in the BPEL Designer. For an overview of the BPEL Designer, see Oracle Fusion Middleware Developer’s Guide for Oracle SOA Suite.</td>
</tr>
</tbody>
</table>

Notice, too, the PartnerSupplierComposite project displays in the Application Navigator.

5. Click the composite.xml tab to view PartnerSupplierComposite composite. SOAP service binding component external_partnersupplier_client in the left swim lane provides the outside world with an entry point into the SOA composite application.
6. Click the WebLogicFusionOrderDemo.jws tab to view the contents of the WebLogicFusionOrderDemo application.

7. Select Save All from the File main menu to save your work.

3.2.3 Task 3: Modify the ExternalPartnerSupplier BPEL Process

Next, you create an assign activity to take the purchase amount and the order date and the current date as input variables to the ExternalPartnerSupplier service. An assign activity transfers data between variables, expressions, and other elements.

1. Click the ExternalPartnerSupplier tab.

2. Create the assign activity:
   a. From the Component Palette, drag an Assign activity below the receiveInput receive activity.
   b. Rename this activity by double-clicking the name underneath the icon. Do not double-click the activity icon itself.
   c. In the edit field, change the name to AssignResponse.
   d. Double-click the assign activity.

The Assign dialog displays.
Creating the WebLogicFusionOrderDemo Application and the PartnerSupplierComposite Composite

3. Assign the purchase amount of the input to the OrderTotal variable for the ExternalPartnerSupplier service:
   a. From the dropdown list, select **Copy Operation**:

   ![Copy Operation dialog](image)

   The Create Copy Operation dialog appears.
   b. On the **From** side, leave the **Type** selection as **Variable** and expand **Variables > inputVariable > payload > client: WarehouseProcessRequest** and select **client: orderTotal**.
   c. On the **To** side, leave the **Type** selection as **Variable** and expand **Variables > outputVariable > payload > client: WarehouseProcessResponse** and select **client: orderTotal**.
   d. Click **OK** to close the Create Copy Operation dialog and return to the Assign dialog.

   The Copy Operation tab in the Assign updates to show the operation you created.

4. Assign the current date to the output deliveryDate variable for the ExternalPartnerSupplier service:
   a. From the dropdown list in the Assign dialog, select **Copy Operation**.

   The Create Copy Operation dialog appears.
   b. On the **From** side, from the **Type** list, select **Expression**.
   c. Select the **XPath Expression Builder** icon.
Creating the WebLogicFusionOrderDemo Application and the PartnerSupplierComposite Composite

The Expression Builder displays.

d. In the **Functions** section in the lower right, from the menu, select **Date Functions**, and then select **current-date** from the menu options.

e. Click **Insert Into Expression**, and then click **OK** to return to the Create Copy Operation dialog.

f. On the **To** side, leave the **Type** selection as **Variable** and expand **Variables > outputVariable > payload > client:WarehouseResponse** and select **client:deliveryDate**.

g. Click **OK** to close the Create Copy Operation dialog and return to the Assign dialog.

The Copy Operation tab in the Assign dialog updates to show the two operations you created.

5. In the Assign dialog, click **OK**.
6. Select Save All from the File main menu to save your work.
7. Click X in the ExternalPartnerSupplier bpel tab to close the process.
8. Click X in the composite.xml tab to close the composite.

The PartnerSupplierComposite composite and ExternalPartnerSupplier BPEL process are now complete.

### 3.2.4 Task 4: Deploy the PartnerSupplierComposite Composite

To deploy the PartnerSupplierComposite composite:

1. In the Application Navigator, right-click PartnerSupplierComposite and select Deploy > PartnerSupplierComposite > to MyAppServerConnection.

The SOA Deployment Configuration Dialog displays.

2. Accept the default settings and click OK.

3. When prompted with the Authorization Request dialog, enter weblogic in the Username field and the password in the Password field.

   In SOA - Log, a series of validations display, followed by:
   
   BUILD SUCCESSFUL
   Total time: nn seconds

### 3.2.5 Task 6: Initiate a Test Instance for the PartnerSupplierComposite Composite

In this task, you initiate a test instance of the PartnerSupplierComposite composite from the Test Web Service page in Fusion Middleware Control to verify the assign activity is working properly.

To initiate a test instance of the PartnerSupplierComposite composite:

1. Start Fusion Middleware Control. See Section 2.3.

2. From the SOA Infrastructure menu, select SOA Administration and select Common Properties.

3. On the Common Properties page, enter the following values to collect data for running instances:

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audit Level</td>
<td>Development</td>
</tr>
<tr>
<td>Capture Composite Instance State</td>
<td>Click to enable.</td>
</tr>
</tbody>
</table>

4. Click Apply to apply changes.

5. Access the Test Web Service page through the following options:
Creating the WebLogicFusionOrderDemo Application and the PartnerSupplierComposite Composite

<table>
<thead>
<tr>
<th>From the SOA Infrastructure Menu...</th>
<th>From the SOA Folder in the Navigator...</th>
<th>From the SOA Composite Menu...</th>
</tr>
</thead>
<tbody>
<tr>
<td>2. Select the Deployed Composites tab.</td>
<td>2. At the top of the page, click Test.</td>
<td></td>
</tr>
<tr>
<td>3. In the Composite section, select PartnerSupplierComposite.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4. At the top of the page, click Test.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The Test Web Service page for initiating an instance appears. This page provides many options for initiating an instance. At a minimum, you must specify the XML payload data to use in the **Input Arguments** section.

The WSDL file and endpoint URL are populated automatically based on the service you selected to test. The endpoint URL is derived from the WSDL and can be overridden to invoke that service at a different location. The port of the current service is displayed.

6. In the **Inputs Arguments** section, enter the input arguments for the Web service:
   a. In the orderItemArray Size field, enter 1 and then click the Array Size icon.

   ![Input Arguments](image1)

   b. Expand orderItems > orderItems in the tree, and enter values for the following fields:
      - OrderTotal
      - quantity
      - productId

   ![Input Arguments](image2)

7. Click Test Web Service.

   The test results appear in the **Response** tab upon completion.

![Response](image3)

The web service invocation was successful. However, there was no response to the invocation from the server.
8. Click **Launch Message Flow Trace** to access the flow trace of the instance.

9. In the Flow Trace window, in the **Trace** section, click the **ExternalPartnerSupplier** instance.

10. In the Flow Trace window for the instance, click the **Flow** tab.

11. Click the **AssignResponse** activity to see the value you entered for **OrderTotal** being copied to the **outputVariable**.

12. Click X or **Close** to dismiss the Activity Details dialog.

This chapter describes how to create the OrderBookingComposite composite of the WebLogic Fusion Order Demo application. This chapter assumes you have performed all the tasks in Chapter 3, "Creating the SOA Application."

This chapter contains the following sections:

- Section 4.1, "About the OrderBookingComposite Composite"
- Section 4.2, "Approaches for Creating OrderBookingComposite"
- Section 4.3, "Creating the OrderBookingComposite Project"
- Section 4.4, "About the OrderProcessor Process"

### 4.1 About the OrderBookingComposite Composite

Chapter 1, "Introduction to the SOA Sample Application," describes the flow of the OrderBookingComposite composite. Figure 4–1 shows the OrderBookingComposite composite in the SOA Composite Editor.
The left swim lane of the SOA Composite Editor contains references that send messages to external services. Table 4–1 describes the services referenced by service components within the OrderBookingComposite composite.
Table 4–1 Exposed Services in OrderBookingComposite Composite

<table>
<thead>
<tr>
<th>Exposed Services</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>orderprocessor_client_ep</td>
<td>This service provides an entry into the OrderProcessor BPEL process, providing an entry point into the OrderBookingComposite composite to process electronic orders from the Store Front module.</td>
</tr>
<tr>
<td>UpdateOrderStatus_ep</td>
<td>This service provides an entry into the UpdateOrderStatus mediator.</td>
</tr>
</tbody>
</table>

The designer (middle section) of the SOA Composite Editor contains service components. Table 4–2 describes the service components used in the OrderBookingComposite composite.

Table 4–2 Service Components in OrderBookingComposite Composite

<table>
<thead>
<tr>
<th>Service Component</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ApprovalHumanTask</td>
<td>Human Task</td>
<td>This component enables a manager to approve or reject an order.</td>
</tr>
<tr>
<td>EvaluatePreferredSupplierRule</td>
<td>Business Rule</td>
<td>This component chooses the shipment supplier based on the lowest bid.</td>
</tr>
<tr>
<td>FulfillOrder</td>
<td>Mediator</td>
<td>This component routes order information to either the USPSShipment file adapter and the FulfillmentBatch JMS adapter.</td>
</tr>
<tr>
<td>InternalWarehouseService</td>
<td>BPEL</td>
<td>This component provides a delivery date (to compete with the price from PartnerSupplierMediator mediator). This process is used to demonstrate invoking an asynchronous process from BPEL.</td>
</tr>
<tr>
<td>OrderPendingEvent</td>
<td>Mediator</td>
<td>This component subscribes to event NewOrderSubmitted from the Oracle Application Development Framework (ADF) Business Component of StoreFrontService, which contains the order ID. OrderPendingEvent consumes the event, transforms it, and passes the order ID to the OrderProcessor BPEL process.</td>
</tr>
<tr>
<td>OrderProcessor</td>
<td>BPEL</td>
<td>This component receives the order ID information, processes the order, and orchestrates all necessary services within the enterprise to complete the order.</td>
</tr>
<tr>
<td>OrderUpdateEventMediator</td>
<td>Mediator</td>
<td>This component subscribes to event OrderUpdateEvent from the UpdateOrderStatus mediator. The OrderUpdateEventMediator transforms the event and passes the order ID from the OrderProcessor BPEL process to StoreFrontService, which sends back the order status and order information, which the mediator transforms.</td>
</tr>
</tbody>
</table>
4.2 Approaches for Creating OrderBookingComposite

When creating a complex composite, you can use the following approaches for building:

- **Top-Down**: You analyze your business processes and identify activities in support of your process. When creating a composite, you define all the SOA components through the SOA Composite Editor. You create all the services first, and then build the BPEL process, referencing the created services.

- **Bottom-Up**: You analyze existing applications and assets to identify those that can be used as services. As you create a BPEL process, you build the services on as-needed basis. This approach works well when IT must react to a change.

---

### Table 4–3 References in OrderBookingComposite Composite

<table>
<thead>
<tr>
<th>Service Component</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreditCardAuthorization</td>
<td>Web Service</td>
<td>This synchronous service provides the credit card type, account number, and purchase amount to the OrderProcessor BPEL process.</td>
</tr>
<tr>
<td>StoreFrontService</td>
<td>Web Service</td>
<td>This synchronous service provides customer ID information to the OrderProcessor BPEL process.</td>
</tr>
<tr>
<td>FulfillmentBatch</td>
<td>JMS Adapter</td>
<td>This adapter provides a JMS queue for storing all fulfillment orders for overnight batch processing. The JMS adapter is used to write the order information to the specified JMS queue.</td>
</tr>
<tr>
<td>PartnerSupplierService</td>
<td>Web Service</td>
<td>This asynchronous service provides the lowest bid for the order from PartnerSupplierComposite composite.</td>
</tr>
<tr>
<td>NotificationService</td>
<td>Web Service</td>
<td>This synchronous service provides an Oracle Messaging Service for notifying the customer of the order.</td>
</tr>
<tr>
<td>USPSShipment</td>
<td>File Adapter</td>
<td>This adapter ships the order using USPS.</td>
</tr>
</tbody>
</table>
For the tutorial, you use the bottom-up approach, so you can learn to build the composite in discrete segments.

4.3 Creating the OrderBookingComposite Project

In this procedure, you create the OrderBookingComposite project and the OrderProcessor BPEL process. This procedure contains the following tasks:

- Task 1: Create the OrderBookingComposite Project
- Task 2: Create the OrderProcessor BPEL Process
- Task 3: Add the ADF Business Components Service Runtime Library

4.3.1 Task 1: Create the OrderBookingComposite Project

To create the OrderBookingComposite project for the WebLogic Fusion Order Demo application:

1. Right-click the WebLogicFusionOrderDemo application name in the Application Navigator and select New Project.
   The New Gallery dialog displays.
2. From either the All Technologies tab or the Current Project Technologies tab, in the Categories tree, select SOA Tier.
3. In the Items list, select SOA Project.
4. Click OK.
   The Create SOA Project dialog appears.
5. Enter the following values:

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Name</td>
<td>OrderBookingComposite</td>
</tr>
<tr>
<td>Directory</td>
<td>Accept the default directory location, MY_FOD_HOME\CompositeServices\OrderBookingComposite. Oracle JDeveloper creates this directory for all the contents of the OrderBookingComposite project.</td>
</tr>
<tr>
<td>Project Technologies</td>
<td>SOA</td>
</tr>
</tbody>
</table>

6. Click Next.
7. In the Configure SOA Settings page, from the Composite Template section, select Composite With BPEL.
8. Click Finish.
   The Create BPEL Process dialog displays.

4.3.2 Task 2: Create the OrderProcessor BPEL Process

Now, continue with the creation of the OrderProcessor BPEL process:

1. In the Create BPEL Process dialog, enter the following values:
2. In the Input field, import the complete schema located in the DEMO_DOWNLOAD_HOME directory.
   
a. In the Input field, click the Browse Input Elements icon.
   
The Type Chooser dialog displays.

b. Click the Import Schema File icon.
   
The Import Schema File dialog displays.

c. Click the Browse Resources icon to the right of the URL field.
   
The SOA Resource Browser displays.

d. Select File System and in the Location section, search for InternalWarehouse.xsd in DEMO_DOWNLOAD_HOME/CompositeServices/OrderBookingComposite/xsd and click OK.

e. In the Import Schema dialog, ensure the InternalWarehouse.xsd now displays in the URL field and the Copy to Project option is selected, and then click OK.
   
The Localized Files dialog displays, prompting you to import the InternalWarehouse.xsd schema file.

f. Deselect the Maintain original directory for imported files option and click OK to import the files.
   
The Type Chooser dialog displays.

g. Expand Project Schema Files > InternalWarehouse.xsd and select WarehouseRequest and then click OK.

3. In the Create BPEL Process dialog, import the elements from the InternalWarehouse.xsd file for the output:
   
a. In the Output field, click the Browse Output Elements icon.
   
The Type Chooser dialog displays.

b. Expand Project Schema Files > InternalWarehouse.xsd and select WarehouseResponse and then click OK.

4. In the Create BPEL Process dialog, click OK.
   
The OrderProcessor BPEL process displays in the designer. Notice, too, the OrderBookingComposite project displays in the Application Navigator.
In Chapter 5, "Creating the First Half of the OrderProcessor BPEL Process," you create services and building blocks for placing an order.

5. Click the **composite.xml** tab to view OrderBookingComposite. SOAP service binding component `orderprocessor_client_ep` in the left swim lane provides the outside world with an entry point into the SOA composite application.

6. Click the **WebLogicFusionOrderDemo.jws** tab to view the contents of the **WebLogicFusionOrderDemo** application.

7. From the **Show** list, select **OrderBookingComposite** to view the contents of the **OrderBookingComposite** application.
About the OrderProcessor Process

8. Select **Save All** from the **File** main menu to save your work.

4.3.3 Task 3: Add the ADF Business Components Service Runtime Library

To add the ADF Business Components service run-time library:

1. In the Application Navigator, right-click **OrderBookingComposite** and select **Project Properties**.

2. Select **Libraries and Classpath**, and from the Libraries and Classpath page, and click **Add Library**.

3. In the Add Library dialog, select **BC4J Service Runtime**, and then click **OK**.

4. In the Libraries and Classpath page, click **OK**.

4.4 About the OrderProcessor Process

The **OrderProcessor** process represents the main flow in the WebLogic Fusion Order Demo application. It sends the order information to the appropriate services at the appropriate times. For example, it contacts the **CreditAuthorizationService** service to check the customer’s credit card, and if the credit card is acceptable, it contacts the internal and external warehouses to get price quotes for the order.

The **OrderProcessor** project is a large project. This chapter begins by giving an overview of the major blocks in the project, and then it goes into detail on how to create each block.

Table 4–4 lists the major blocks in the **OrderProcessor** process:

<table>
<thead>
<tr>
<th>Block</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>receiveInput</td>
<td>Receive activity</td>
<td>This activity receives the order ID for incoming requests.</td>
</tr>
</tbody>
</table>
About the OrderProcessor Process

Creating the OrderBookingComposite Composite

Table 4–4 (Cont.) Major Blocks in the OrderProcessor Process

<table>
<thead>
<tr>
<th>Block</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scope_RetrieveOrder</td>
<td>Scope</td>
<td>This scope retrieves order information from the database. It uses a bind entity activity to point to order data in an Oracle Application Development Framework (ADF) Business Component data services provider.</td>
</tr>
<tr>
<td>Scope_RetrieveCustomerForOrder</td>
<td>Scope</td>
<td>This scope calls the StoreFrontService service to retrieve customer information.</td>
</tr>
<tr>
<td>Scope_AuthorizeCreditCard</td>
<td>Scope</td>
<td>This scope verifies that the customer has acceptable credit using the CreditCardAuthorizationService service.</td>
</tr>
<tr>
<td>Scope_CheckApprovalLimit</td>
<td>Scope</td>
<td>This scope does the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Initiates the RequiresApproval business rule that evaluates the amount of the order to determine whether an order must be approved by a manager.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ For an order where approval is required, uses a switch to initiate the ApprovalHumanTask human task for a manager to approve or not approve the order.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For approved orders, the OrderProcessor process continues with the rest of the activities. For rejected orders, the process throws a fault and does not continue.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ For orders where approval is not required, the switch is not initiated.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Sets the variables for the price amount and credit card status used by the RequiresApprovalRule business rule.</td>
</tr>
<tr>
<td>Scope_RetrieveQuotes</td>
<td>Scope</td>
<td>This scope sends order information to two suppliers, an external partner warehouse and an internal warehouse, and the warehouses return their bids for the orders.</td>
</tr>
<tr>
<td>Scope_SelectPreferredSupplier</td>
<td>Scope</td>
<td>This scope initiates the SelectPreferredSupplier business rule for selecting a shipping supplier with the lowest bid.</td>
</tr>
<tr>
<td>Scope_FulfillOrder</td>
<td>Scope</td>
<td>This scope calls the FulfillOrder mediator component, which determines the shipping method for the order.</td>
</tr>
<tr>
<td>Scope_UpdateStatusToComplete</td>
<td>Scope</td>
<td>This scope assigns a final status of complete to the order.</td>
</tr>
<tr>
<td>Scope_NotifyCustomerOfCompletion</td>
<td>Scope</td>
<td>This scope uses the Oracle User Messaging Service to send an email to the customer who placed the order.</td>
</tr>
<tr>
<td>callbackClient</td>
<td>Invoke activity</td>
<td>This invoke activity notifies the client that it is done.</td>
</tr>
</tbody>
</table>
Figure 4–2 shows the OrderProcessor process in the BPEL Designer of the Oracle JDeveloper with the blocks minimized. Exercises in Chapter 5 and Chapter 6 expand the blocks to show their contents and describe how to create the blocks.
Figure 4–2  Minimized View of the Blocks in OrderProcessor
5

Creating the First Half of the OrderProcessor BPEL Process

This chapter describes how to create the first half of the OrderProcessor BPEL process for the OrderBookingComposite composite. This chapter assumes you have performed all the tasks from Chapter 3 through Chapter 4.

This chapter contains the following sections:

- Section 5.1, "Overview of Tasks for Creating the First Half of OrderProcessor"
- Section 5.2, "Copying Services Used by the OrderProcessor BPEL Process"
- Section 5.3, "Adding the StoreFrontService Service"
- Section 5.4, "Wiring the OrderProcessor BPEL Process to the StoreFrontService Service"
- Section 5.5, "Creating the gOrderInfoVariable Variable"
- Section 5.6, "Creating the Scope_RetrieveOrder Scope"
- Section 5.7, "Creating the Scope_RetrieveCustomerForOrder Scope"
- Section 5.8, "Creating CreditCardAuthorizationService Service"
- Section 5.9, "Creating the Scope_AuthorizeCreditCard Scope"
- Section 5.10, "Creating Credit Card Authorization Service for the Scope_AuthorizeCreditCard"
- Section 5.11, "Creating the RequiresApprovalRule Business Rule"
- Section 5.12, "Adding the Switch_ApprovalRequired Switch to the Scope_CheckApprovalLimit Scope"

5.1 Overview of Tasks for Creating the First Half of OrderProcessor

Table 5–1 lists and describes the tasks for creating the first half of the OrderProcessor BPEL process for the OrderBookingComposite composite.
Copy the service definitions you reference later during the modification of the OrderProcessor BPEL process.

### Table 5–1  Tasks for Creating the First Half of the OrderProcessor BPEL Process for OrderBookingComposite

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy Services Used by the OrderProcessor BPEL Process</td>
<td>Copy the service definitions you reference later during the modification of the OrderProcessor BPEL process.</td>
<td>Section 5.2</td>
</tr>
<tr>
<td>Add the StoreFrontService Service</td>
<td>Reference the StoreFrontService service.</td>
<td>Section 5.3</td>
</tr>
<tr>
<td>Wire the OrderProcessor BPEL Process to the StoreFrontService Service</td>
<td>Connect StoreFrontService to the OrderProcessor BPEL process.</td>
<td>Section 5.4</td>
</tr>
<tr>
<td>Create the gOrderInfoVariable Variable</td>
<td>Create variable gOrderInfoVariable as input for when an order is placed. To create this variable, you create it as an entity variable.</td>
<td>Section 5.5</td>
</tr>
<tr>
<td>Create the Scope_RetrieveOrder Scope</td>
<td>Create the Scope_RetrieveOrder scope to obtain the order ID using the gOrderInfoVariable entity variable.</td>
<td>Section 5.6</td>
</tr>
<tr>
<td>Create the Scope_RetrieveCustomerForOrder Scope</td>
<td>Create the Scope_RetrieveCustomerForOrder scope to call the StoreFrontService service to retrieve customer information.</td>
<td>Section 5.7</td>
</tr>
<tr>
<td>Create CreditCardAuthorizationService Service</td>
<td>Reference the CreditCardAuthorizationService service, which checks whether the customer's credit card is valid.</td>
<td>Section 5.8</td>
</tr>
<tr>
<td>Create the Scope_AuthorizeCreditCard Scope</td>
<td>Create the Scope_AuthorizeCreditCard scope to initiate the CreditCardAuthorizationService service to retrieve customer information.</td>
<td>Section 5.9</td>
</tr>
<tr>
<td>Create Catch Branches for the Scope_AuthorizeCreditCard Scope</td>
<td>Add catch branches to the Scope_AuthorizeCreditCard scope for orders in which the credit card number is not provided or the credit type is not valid.</td>
<td>Section 5.10</td>
</tr>
<tr>
<td>Create the RequiresApprovalRule Business Rule</td>
<td>Create a business rule activity to specify the RequiresApprovalRule business rule. This rule specifies that if the order total is $2,000 or more, then a manager's approval is required. Another activity uses the output from the business rule to either automatically approve the order or use a human task to obtain manager approval.</td>
<td>Section 5.11</td>
</tr>
<tr>
<td>Create the Switch_ApprovalRequired Switch to the Scope_CheckApprovalLimit Scope</td>
<td>For an order that requires manual approval because the order total is $2,000 or more, you create the SwitchApprovalRequired switch in the ScopeCheckApprovalLimit scope with a &lt;case&gt; branch that passes control to the ApprovalHumanTask human task activity. This human task enables a manager to approve or reject the orders. This switch does not apply to orders that do not require manual approval.</td>
<td>Section 5.12</td>
</tr>
</tbody>
</table>

5.2 Copying Services Used by the OrderProcessor BPEL Process

Copy the service definitions you reference later during the modification of the OrderProcessor BPEL process:
1. Copy directory services from `DEMO_DOWNLOAD_HOME\CompositeServices\OrderBookingComposite` to directory `MY_FOD_HOME\CompositeServices\OrderBookingComposite`.

2. In the Application Navigator, select `OrderBookingComposite` and then click the Refresh icon.

The `OrderBookingComposite` folder in Oracle JDeveloper updates with the services folder.

5.3 Adding the StoreFrontService Service

The `StoreFrontService` service contains order and customer information. Perform the following tasks to reference this synchronous service:

- Task 1: Copy the WSDL Needed for StoreFrontService
- Task 2: Create a Web Service for StoreFrontService

5.3.1 Task 1: Copy the WSDL Needed for StoreFrontService

To copy the WSDL for the `StoreFrontService` service:

1. Copy `StoreFrontServiceRef.wsdl` from `DEMO_DOWNLOAD_HOME\CompositeServices\OrderBookingComposite` to directory `MY_FOD_HOME\CompositeServices\OrderBookingComposite`.

2. In the Application Navigator, select `OrderBookingComposite` and then click the Refresh icon.

5.3.2 Task 2: Create a Web Service for StoreFrontService

To create a Web service for the `StoreFrontService` service:

1. Click the `composite.xml` tab to view the SOA Composite Editor again.

2. From the Component Palette, drag a `Web Service` from the `Service Adapters` list into the right swim lane (`External References`) of the SOA Composite Editor.
Adding the StoreFrontService Service

The Create Web Service dialog appears.

3. Enter and select the following values:

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>StoreFrontService</td>
</tr>
<tr>
<td>Type</td>
<td>Reference</td>
</tr>
<tr>
<td>WSDL URL</td>
<td>Click the <strong>Find existing WSDLs</strong> icon and select StoreFrontServiceRef.wsdl from <strong>MY_HOME\OrderBookingComposite</strong></td>
</tr>
</tbody>
</table>

4. Accept the other defaults, and then click **OK**.

The StoreFrontService service displays in the SOA Composite Editor.
If you receive an error, then it is likely you did not add the BC4J Runtime Service library the OrderBookingComposite project. See Section 4.3.3 to add the library.

5.4 Wiring the OrderProcessor BPEL Process to the StoreFrontService Service

To wire (connect) the StoreFrontService service to the OrderProcessor BPEL service component:

1. Drag a wire from the OrderProcessor BPEL process interface to the StoreFrontService reference handle.

2. Click Source at the bottom of the visual editor to review the wiring:

   <wire>
   <source.uri>OrderProcessor/StoreFrontService</source.uri>
   <target.uri>StoreFrontService</target.uri>
   </wire>

3. Click Design at the bottom of the visual editor.

4. Click the OrderProcessor.bpel tab to view the BPEL process again.

   The StoreFrontService service displays in the BPEL Designer.
5. Select **Save All** from the **File** main menu to save your work.

### 5.5 Creating the gOrderInfoVariable Variable

In this task, you create variable `gOrderInfoVariable` as input for when an order is placed. The letter `g` is used to distinguish this variable as a global variable that can be used throughout the BPEL process. This tutorial requests that you create global variables with a letter `g` prefix and local variables for individual scopes with a letter `l` prefix. You can use a local variable only within a scope.

In previous releases, variables and messages exchanged within a BPEL business process were disconnected payload (a snapshot of data returned by a Web service) placed into an XML structure. In some cases, the user required this type of fit. In other cases, this fit presented challenges.

For this release, the entity variable can be used with an Oracle ADF Business Component data provider service using SDO-based data.

To create an entity variable for the purchase order ID and select the `StoreFrontService` as a partner link to invoke the Oracle ADF Business Component application:

1. In the canvas workspace for the **OrderProcessor** BPEL process, click the **Variables** icon.

The Variables dialog displays.
2. Click the Create icon to add a variable.

   The Create Variable dialog displays.

3. In the Name field, enter gOrderInfoVariable. Use the letter g to distinguish this variable as a global variable that can be used throughout the process.

4. In the Type section, select Element and then select the Search icon to the right of the Element field.

   The Type Chooser dialog appears with a list of available services.

5. Expand Project Schema Files > OrderInfoVOSDO.xsd > orderInfoVOSDO.

   OrderId represents the order in the Oracle ADF Business Component of the StoreFrontService.

6. Select orderInfoVOSDO, and then click OK.

7. In the Create Variable dialog, click the Entity Variable check box and select the Search icon to the right of the Partner Link field.

   The Partner Link Chooser window displays.

8. Expand Process > Partner Links and select StoreFrontService, and then click OK.

   The Create Variable dialog shows the element and service information.
9. Click **OK** in the Create Variable dialog.

The Variables dialog updates with the `gOrderInfoVariable` entity variable.

10. Click **OK** in the Variables dialog.

### 5.6 Creating the Scope_RetrieveOrder Scope

The `Scope_RetrieveOrder` scope uses a bind entity activity to obtain the order ID using the `gOrderInfoVariable` entity variable.
A scope activity does not actually execute or do anything, it simply holds other activities. Scopes are analogous to curly braces in Java. You can use them to break up your process into logical chunks.

To create this scope, perform these tasks:

- **Task 1: Add the Scope_RetrieveOrder Scope**
- **Task 2: Create findOrderById Bind Entity Activity**

### 5.6.1 Task 1: Add the Scope_RetrieveOrder Scope

To create the `Scope_RetrieveOrder` scope:

1. From the Component Palette, drag a `Scope` activity below the `receiveInput` activity.

2. Rename this activity by double-clicking the name underneath the icon. Do not double-click the activity icon itself.

3. In the edit field, change the name to `Scope_RetrieveOrder`.

4. Click the `Expand (+)` icon to expand the `Scope_RetrieveOrder` scope.

### 5.6.2 Task 2: Create findOrderById Bind Entity Activity

To create a key to point to the data in the Oracle ADF Business Component data provider service:
1. From the Component Palette, drag a **Bind Entity** activity into the **Scope_RetrieveOrder** scope.

2. Double-click the **Bind Entity** activity.

   The Bind Entity window displays.

3. In the **Name** field, enter `findOrderById`.

4. Click the **Search** icon next to the **Entity Variable** field.

   The Browse Variables dialog appears.

5. Select the `gOrderInfoVariable` variable you created in Section 5.5, and then click **OK**.

6. In the **Unique Keys** section of the Bind Entity window, click the **Create** icon to create a key for retrieving the order ID from the database.

   The Specify Key dialog appears.

7. Configure the following settings to define the binding key:
Creating the Scope_RetrieveCustomerForOrder Scope

<table>
<thead>
<tr>
<th>Element</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key Local Part</td>
<td>1. Click the Browse Entity Variable icon. It is the icon to the right of the Key Local Part field. The Browse Entity Key dialog appears.</td>
</tr>
<tr>
<td></td>
<td>2. Expand Variables &gt; gOrderInfoVariable &gt; ns4:orderInfoVOSDO and select element ns4:OrderId. Do not select the OrderId key. The namespace number values (for example, ns1, ns2) can vary.</td>
</tr>
<tr>
<td></td>
<td>3. Click OK.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Key Namespace URI</th>
<th>Leave the default for the namespace URI for the key.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key Value</td>
<td>1. Click the Expression Builder icon.</td>
</tr>
<tr>
<td></td>
<td>The Expression Builder dialog appears.</td>
</tr>
<tr>
<td></td>
<td>2. In the BPEL Variable section, expand Variables &gt; payload &gt; ns4:WarehouseRequest and select ns4:orderId.</td>
</tr>
<tr>
<td></td>
<td>3. Click Insert Into Expression.</td>
</tr>
<tr>
<td></td>
<td>4. Click OK in the Specify Key dialog.</td>
</tr>
</tbody>
</table>

8. Click OK to close the Specify Key dialog.

A name-pair value appears in the Unique Keys table.

9. Click OK to close the Bind Entity window.

When the bind entity activity is executed at run time, the gOrderInfoVariable entity variable is ready to be used. Otherwise, a run-time fault results.

10. Click the Collapse (-) icon to minimize the Scope_RetrieveOrder scope.

5.7 Creating the Scope_RetrieveCustomerForOrder Scope

The Scope_RetrieveCustomerForOrder scope calls the StoreFrontService service to retrieve customer information. It assigns the customer ID from global variable gOrderInfoVariable to local variable lFindCustomerInfo_
Creating the Scope_RetrieveCustomerForOrder Scope

InputVariable for the scope. The scope then uses an invoke activity to call the StoreFrontService service. The invoke activity provides the lFindCustomerInfo_InputVariable variable as input to the service, and the StoreFrontService service returns information about the customer, such as the customer name and email address, back to the BPEL process through the gCustomerInfoVariable global variable.

Figure 5–2 Scope_RetrieveCustomerForOrder

To create this scope, perform the following tasks:

■ Task 1: Add the Scope_RetrieveCustomerForOrder Scope
■ Task 2: Create the InvokeCustomerService Activity
■ Task 3: Create the AssignCustomerId Activity
■ Task 4: Deploy the OrderBookingComposite Composite
■ Task 5: Deploy the OrderSDOComposite Composite
■ Task 6: Initiate a Test Instance for the OrderBookingComposite Composite

5.7.1 Task 1: Add the Scope_RetrieveCustomerForOrder Scope

To create the Scope_RetrieveCustomerForOrder scope:

1. From the Component Palette, drag a Scope activity below the Scope_RetrieveOrder activity.
2. Rename this activity by double-clicking the name underneath the icon. Do not double-click the activity icon itself.
3. In the edit field, change the name to Scope_RetrieveCustomerForOrder.
4. Click the Expand (+) icon to expand the Scope_RetrieveCustomerForOrder scope.

5.7.2 Task 2: Create the InvokeCustomerService Activity

An invoke activity invokes a service and passes it data, and in this case, wait for a response from the service with the return data. To call the StoreFrontService service, you create an invoke activity:

1. From the Component Palette, drag an Invoke activity into the Scope_RetrieveCustomerForOrder scope.
2. Rename this activity by double-clicking name underneath the icon. Do not double-click the invoke icon itself.

3. In the edit field, change the name to `InvokeFindCustomer`.

4. Drag the mouse from the right side of `InvokeFindCustomer` to the `StoreFrontService` partner link.

The Edit Invoke dialog appears and is automatically filled in with the following information:

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td><code>InvokeFindCustomer</code></td>
</tr>
<tr>
<td>Partner Link</td>
<td><code>StoreFrontService</code></td>
</tr>
<tr>
<td>Operation</td>
<td><code>findCustomerInfoV01</code></td>
</tr>
</tbody>
</table>

5. In the *Operation* field, change the selection to `findCustomerInfoV01CustomerInfoVOCriteria`.

6. Click the *Automatically Create Input Variable* icon. It is the first icon to the right of the *Input Variable* field.

The Create Variable dialog appears for the input variable. This variable provides input data to the `StoreFrontService` service, namely the ID of the customer.

7. Enter and select the following values:

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td><code>lFindCustomerInfo_InputVariable</code></td>
</tr>
</tbody>
</table>

Use the letter `l` to distinguish this variable as a local variable that can be used only within this scope. This tutorial requests that you create global variables with a letter `g` prefix and local variables for individual scopes with a letter `l` prefix. You can use a local variable only within a scope.
5.7.3 Task 3: Create the AssignCustomerActivity

In this task, you create an assign activity to take the customer ID and send it to the input variable for the StoreFrontService service.

1. From the Component Palette, drag an Assign activity above the InvokeFindCustomer invoke activity.
2. Rename this activity by double-clicking name underneath the icon. Do not double-click the assign icon itself.
3. In the edit field, enter AssignCustomerActivity.
4. Double-click the AssignCustomerActivity activity.
   The Assign dialog displays.
5. From the dropdown list, select Copy Operation:

   ![Copy Operation dialog]

   The Create Copy Operation dialog appears.
6. On the From side, expand Variables > gOrderInfoVariable > ns4:orderInfoVOSDO > ns4:CustomerId. The namespace number values (for example, ns1, ns2) can vary.

7. On the To side, expand Scope - Scope_RetrieveCustomerForOrder > Variables > lFindCustomerInfo_InputVariable > parameters > ns4:findCustomerInfoVO1CustomerInfoVOCriteria and select ns4:CustId. The namespace number value can vary.

8. Click OK to close the Create Copy Operation dialog.

The Copy Operation tab in the Assign updates to show the operation you created.

9. In the Assign dialog, click OK.

10. Click the Collapse (-) icon to minimize the Scope_RetrieveCustomerForOrder scope.

11. Select Save All from the File main menu to save your work.

### 5.7.4 Task 4: Deploy the OrderBookingComposite Composite

To deploy the OrderBookingComposite composite:

1. In the Application Navigator, right-click OrderBookingComposite and select Deploy > OrderBookingComposite > to MyAppServerConnection. You created the MyAppServerConnection connection in Section 1.2.4, "Task 4: Create a Connection to an Oracle WebLogic Server."

   The SOA Deployment Configuration Dialog displays.

2. Accept the default settings and click OK.

3. When prompted with the Authorization Request dialog, enter weblogic in the Username field and the password in the Password field.

   In SOA - Log, a series of validations display, followed by:

   BUILD SUCCESSFUL
   Total time: nn seconds
5.7.5 Task 5: Deploy the OrderSDOComposite Composite

To initiate a test instance of the OrderBookingComposite composite, you must deploy a service using the StoreFrontService.wsdl. If you performed the tasks in Section 2.1.2 and have the Store Font module currently running, then you can use it to test the OrderBookingComposite composite. You can proceed to Section 5.7.6, "Task 6: Initiate a Test Instance for the OrderBookingComposite Composite." If you do not have the Store Front module currently running, then deploy the OrderSDOComposite composite, available from the sample application.

To deploy the OrderSDOComposite composite:

1. In the Application Navigator, select WebLogicFusionOrderDemo for the sample application in the DEMO_DOWNLOAD_HOME directory.

2. In the Application Navigator, right-click OrderSDOComposite and select Deploy > OrderSDOComposite > to MyAppServerConnection.

   The SOA Deployment Configuration Dialog displays.

3. Accept the default settings and click OK.

4. When prompted with the Authorization Request dialog, enter weblogic in the Username field and the password in the Password field.

   In SOA - Log, a series of validations display, followed by:

   BUILD SUCCESSFUL
   Total time: nn seconds

5. In the Application Navigator, select WebLogicFusionOrderDemo for the application you currently building in the MY_FOD_HOME directory.

5.7.6 Task 6: Initiate a Test Instance for the OrderBookingComposite Composite

In this task, you can initiate a test instance of the OrderBookingComposite composite in one of two ways:

- You can use the Store Front module by submitting an order, similarly to the first order described in Section 2.2 and monitor the order instance described in Section 2.4. The order should progress through the Scope_RetrieveCustomerForOrder scope.

- Use the OrderSDOComposite composite by initiating a test instance of the OrderSDOComposite composite from the Test Web Service page in Fusion Middleware Control. Use the steps described next.

To initiate a test instance of the OrderSDOComposite composite:

1. Access the Test Web Service page in Fusion Middleware Control through the following options:

   From the SOA Infrastructure Menu...
   1. Select Home.

   From the Deployed Composites tab...
   2. Select the Deployed Composites tab.

   From the Composite section...
   3. In the Composite section, select OrderBookingComposite.

   At the top of the page, click Test.

   From the SOA-Infra...
   1. Under soa-infra, select OrderBookingComposite.

   From the Test Service tab...
   2. At the top of the page, click Test.
2. In the Inputs Arguments section of the Test Web Service page, in the orderID field, enter an ID under 1000.

3. Click Test Web Service.

   The test results appear in the Response tab upon completion.

4. Click Launch Message Flow Trace to access the flow trace of the instance.

5. In the Flow Trace window, in the Trace section, click the OrderProcessor BPEL process.

6. In the Flow Trace window for the instance, click the Flow tab.

7. Click the various activities to see the flow through the Scope_RetrieveCustomerForOrder scope.

8. Click X or Close to dismiss the Activity Details dialog.


---

5.8 Creating CreditCardAuthorizationService Service

The CreditAuthorizationService service checks whether the customer's credit card is valid. You create it by creating a Web service based on a WSDL from the FusionOrderDemo_R1.zip.

- Task 1: Copy WSDL File Needed for CreditCardAuthorizationService
- Task 2: Create a Web Service for CreditCardAuthorizationService

Later, in Section 5.8, "Creating CreditCardAuthorizationService Service," you create a scope for the OrderProcessor BPEL process to call this service.

---

5.8.1 Task 1: Copy WSDL File Needed for CreditCardAuthorizationService

Copy CreditCardAuthorizationService.wsdl from directory DEMO_DOWNLOAD_HOME\CompositeServices\OrderBookingComposite to directory MY_FOD_HOME\CompositeServices\OrderBookingComposite.

5.8.2 Task 2: Create a Web Service for CreditCardAuthorizationService

In Section 5.3.2, "Task 2: Create a Web Service for StoreFrontService," you created a reference to the Web service from the SOA Composite Editor. In this task, you create the reference from the BPEL Designer.

To create a Web service for the CreditCardAuthorizationService service:

1. From the Component Palette, drag a Partner Link (Web Service/Adapter) from the BPEL Services list into the right swim lane of the BPEL Designer.

   The Create Partner Link dialog appears.

2. Enter and select the following values:

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>CreditCardAuthorizationService</td>
</tr>
<tr>
<td>WSDL file</td>
<td>Browse and select CreditCardAuthorizationService.wsdl from MY_FOD_HOME\OrderBookingComposite.</td>
</tr>
</tbody>
</table>
Creating the Scope_AuthorizeCreditCard Scope

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partner Link Type</td>
<td>Leave the default as CreditCardAuthorizationService.</td>
</tr>
<tr>
<td>Partner Role</td>
<td>CreditAuthorizationPort</td>
</tr>
<tr>
<td>My Role</td>
<td>Leave the default as Not Specified, since this service is synchronous</td>
</tr>
</tbody>
</table>

3. Click OK.

The CreditCardAuthorizationService service displays in the right swim lane.

If you click the composite.xml tab, you can see the CreditCardAuthorizationService service automatically propagates to the SOA Composite Editor.

![Composite Editor Screenshot]

**5.9 Creating the Scope_AuthorizeCreditCard Scope**

The Scope_AuthorizeCreditCard scope calls the CreditCardAuthorizationService service to retrieve customer information. It assigns the order total, credit card type, and the account number from global variable gOrderInfoVariable to local variable lCreditCardInput for the scope. The scope then uses an invoke activity to call the CreditCardAuthorizationService service. The invoke activity provides the lCreditCardInput variable as input to the service, and the CreditCardAuthorizationService service returns the status back to the BPEL process through the lCreditCardCardOutput local variable. This switch activity checks the results of the credit card validation.
Creating the Scope_AuthorizeCreditCard Scope

To create this scope, perform the following tasks:

- **Task 1: Add the Scope_AuthorizeCreditCard Scope**
- **Task 2: Create the InvokeCheckCredit Invoke Activity**
- **Task 3: Create the Assign_CreditCheckInput Activity**
- **Task 4: Create Switch Activity**

### 5.9.1 Task 1: Add the Scope_AuthorizeCreditCard Scope

To create the `Scope_AuthorizeCreditCard` scope:

1. Back in the `OrderProcessor.bpel` tab, from the Component Palette, drag a `Scope` activity from the `Component Palette` section below the `Scope_RetrieveCustomerForOrder` scope.
2. Rename this activity by double-clicking the name underneath the icon. Do not double-click the activity icon itself.
3. In the edit field, change the name to `Scope_AuthorizeCreditCard`.
4. Click the `Expand (+)` icon to expand the `Scope_AuthorizeCreditCard` scope.

### 5.9.2 Task 2: Create the InvokeCheckCredit Invoke Activity

To create an invoke activity to call `CreditCardAuthorizationService` service:

1. From the Component Palette, drag an `Invoke` activity into the `Scope_AuthorizeCreditCard` scope.
2. Rename this activity by double-clicking name underneath the icon. Do not double-click the invoke icon itself.
3. In the edit field, change the name to `InvokeCheckCreditCard`.
4. Drag the mouse from the right side of `InvokeCheckCreditCard` to the `CreditCardAuthorizationService` partner link.

The Edit Invoke dialog appears and is automatically filled in with the following information:
5. Click the **Automatically Create Input Variable** icon. It is the first icon to the right of the **Input Variable** field.

The Create Variable dialog appears for the input variable. This variable provides input data to `CreditCardAuthorizationService` service.

6. Enter and select the following values:

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>lCreditCardInput</td>
</tr>
<tr>
<td>Global Variable</td>
<td>Local Variable</td>
</tr>
</tbody>
</table>

7. Click **OK** to close the Create Variable dialog.

The Edit Invoke dialog populates with the variable in the **Input Variable** field.

8. Click the **Automatically Create Output Variable** icon. It is the first icon to the right of the **Output Variable** field.

The Create Variable dialog appears for the output variable. This variable returns status from `CreditCardAuthorizationService` service.

9. Enter and select the following values:

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>lCreditCardOutput</td>
</tr>
<tr>
<td>Global Variable</td>
<td>Local Variable</td>
</tr>
</tbody>
</table>

10. Click **OK** to close the Create Variable dialog.

The Edit Invoke dialog populates with the variable in the **Output Variable** field.

11. In the Edit Invoke dialog, click **OK** to save the settings.

### 5.9.3 Task 3: Create the Assign_CreditCheckInput Activity

Next, you create an assign activity to take the credit card type, credit card number, and purchase amount and assign it to the input variable for the `CreditAuthorizationService` service.

1. Create an assign activity to assign data to the input variables for the `CreditAuthorizationService` service:
   a. From the Component Palette, drag an **Assign** activity above the `InvokeCheckCreditCard` invoke activity.
   b. Rename this activity by double-clicking name underneath the icon. Do not double-click the assign icon itself.
   c. In the edit field, enter `Assign_CreditCheckInput`. 

d. Double-click the **Assign_CreditCheckInput** activity.
   The Assign dialog displays.

### 2. Assign an input variable for the purchase amount to the **CreditCardAuthorizationService** service:

a. From the dropdown list, select **Copy Operation**.
   The Create Copy Operation dialog appears.

b. Select the following values:

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>From</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Variable</td>
</tr>
<tr>
<td>Variable</td>
<td>Expand Variables &gt; gOrderInfoVariable &gt; ns4:orderInfoVOSDO and select ns4:OrderTotal. Note: The namespace number values (for example, ns1, ns2) can vary.</td>
</tr>
<tr>
<td>To</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Variable</td>
</tr>
<tr>
<td>Variable</td>
<td>Expand Scope - Scope_AuthorizeCreditCard &gt; Variables &gt; lCreditCardInput &gt; Authorization &gt; ns8:AuthInformation and select ns8:PurchaseAmount.</td>
</tr>
</tbody>
</table>

c. Click **OK** to close the Create Copy Operation dialog and return to the Assign dialog.
   The Copy Operation tab in the Assign dialog updates to show the copy operation.

### 3. Assign an input variable for the type of credit card to the **CreditCardAuthorizationService** service:

a. From the dropdown list, select **Copy Operation**.
   The Create Copy Operation dialog appears.

b. Select the following values:

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>From</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Variable</td>
</tr>
<tr>
<td>Variable</td>
<td>Expand Variables &gt; gOrderInfoVariable &gt; ns4:orderInfoVOSDO and select ns4:CardTypeCode. Note: The namespace number values (for example, ns1, ns2) can vary.</td>
</tr>
<tr>
<td>To</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Variable</td>
</tr>
<tr>
<td>Variable</td>
<td>Expand Scope - Scope_AuthorizeCreditCard &gt; Variables &gt; lCreditCardInput &gt; Authorization &gt; ns8:AuthInformation and select ns8:CCType.</td>
</tr>
</tbody>
</table>
c. Click OK to close the Create Copy Operation dialog and return to the Assign dialog.

The Copy Operation tab in the Assign dialog updates to show the copy operation.

4. Assign an input variable for the credit card account number to the CreditCardAuthorizationService service:
   a. From the dropdown list, select Copy Operation.

   The Create Copy Operation dialog appears.
   b. Select the following values:

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>From</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Variable</td>
</tr>
<tr>
<td>Variable</td>
<td>Expand Variables &gt; gOrderInfoVariable &gt; ns4:orderInfoVOSDO &gt; ns4:AccountNumber. Note: The namespace number values (for example, ns1, ns2) can vary.</td>
</tr>
<tr>
<td>To</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Variable</td>
</tr>
<tr>
<td>Variable</td>
<td>Expand Scope - Scope_AuthorizeCreditCard &gt; Variables &gt; lCreditCardInput &gt; Authorization &gt; ns8:AuthInformation and select ns8:CCNumber.</td>
</tr>
</tbody>
</table>

c. Click OK to close the Create Copy Operation dialog and return to the Assign dialog.

The Copy Operation tab in the Assign dialog updates to show three copy operations.

5. In the Assign dialog, click OK.

6. Select Save All from the File main menu to save your work.
5.9.4 Task 4: Create Switch Activity

To create the switch activity to check the results of the credit card validation:

1. From the Component Palette, drag a **Switch** activity below the **InvokeCheckCreditCard** invoke activity.
2. Rename this activity by double-clicking name underneath the icon. Do not double-click the assign icon itself.
3. In the edit field, enter **Switch_EvaluateCCResult**.
4. Click the **Expand (+)** icon to expand the switch.
5. Specify the conditions for the **<case>** branch to handle cases where a customer’s credit card is not valid. This information is stored in the **lCreditCardOutput** variable.
   a. Double-click the title bar of the **<case>** box to display the Switch Case dialog.
   b. In the Switch Case dialog, click the **XPath Expression Builder** icon above the **Expression** box to display the Expression Builder dialog.
   c. In the Expression Builder dialog, in the **BPEL Variables** box, select **Scope - Scope_AuthorizeCreditCard > Variables > lCreditCardOutput > status** and select **ns8:status**. The namespace number values (for example, **ns1, ns2**) can vary.

   The Content Preview box shows what to insert. For example:
   
   `bpws:getVariableData('lCreditCardOutput','status','/ns8:status')`
   
   d. Click **Insert Into Expression**.
   
   The Expression box updates with the three parameters.
   
   e. Append `!= 'APPROVED'` to the expression in the **Expression** box so that the expression looks like this:

   `bpws:getVariableData('lCreditCardOutput','status','/ns6:status') != 'APPROVED'`

   f. Click **OK** to close the Expression Builder.
   
   g. In the Switch Case dialog, click **OK**.
6. Remove the unneeded <otherwise> branch.
   a. Right-click the <otherwise> branch and select Delete from the menu.
   b. When prompted to remove the branch, click Yes.

7. Create the Throw activity in the remaining <case> branch.
   For those orders not approved, this activity throws a fault called Throw_Fault_CC_Denied. The OrderProcessor process terminates after executing this throw activity.
   a. From the Component Palette, drag a Throw activity into the <case> branch.
   b. Double-click the new throw activity.
      The Throw activity dialog displays.
   c. Enter the following values:

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Throw_Fault_CC_Denied</td>
</tr>
<tr>
<td>Namespace</td>
<td><a href="http://www.globalcompany.example.com/ns/OrderBookingService">http://www.globalcompany.example.com/ns/OrderBookingService</a></td>
</tr>
<tr>
<td>Local Part</td>
<td>OrderProcessorFault</td>
</tr>
<tr>
<td>Fault Variable</td>
<td>Do not enter a value</td>
</tr>
</tbody>
</table>

   d. Click OK.

8. Click the Collapse (-) icon to minimize the switch.

9. Select Save All from the File main menu to save your work.

5.10 Creating Catch Branches for the Scope_AuthorizeCreditCard

Add catch branches to the Scope_AuthorizeCreditCard scope for orders in which the credit card number is not provided or the credit type is not valid. Figure 5–4 shows the catches for the scope.
To create the catches for this scope, perform the following tasks:

- **Task 1: Modify the OrderProcessor.wsdl File for the gOrderProcessorFaultVariable Variable**
- **Task 2: Create the gOrderProcessorFaultVariable Variable**
- **Task 3: Add Catch Branches to the Scope_AuthorizeCreditCard**

### 5.10.1 Task 1: Modify the OrderProcessor.wsdl File for the gOrderProcessorFaultVariable Variable

You create the gOrderProcessorFaultVariable variable as input for the branches in the next task. This variable uses the OrderProcessorFault element from the OrderProcessor.wsdl file, which you must create in the OrderProcessor.wsdl file.

To create this variable:

1. From the Application Navigator, double-click **OrderProcessor.wsdl**.
   The WSDL Editor displays, which is a specialized schema-driven editor for editing WSDL documents.
2. Click the **Source** tab and add the following definition to the `wsdl:definitions` section.
   ```
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   ```
3. Click the **Design** tab.
4. Create the **OrderProcessorFault** message:
   a. Click the **Expand (+)** icon in the **Messages** section.
b. Click the Add icon to add a new message. The Create Message dialog displays.

c. In the Message Name field, enter OrderProcessorFault, and then click OK.

d. Select OrderProcessorFault in Messages.

e. Select OrderProcessorFault and then select part in the right side WSDL component palette.

The Create Part dialog displays.

f. Enter and select following values:

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Part Name</td>
<td>code</td>
</tr>
<tr>
<td>Reference Type</td>
<td>type</td>
</tr>
<tr>
<td>Part Name</td>
<td>xsd:string</td>
</tr>
</tbody>
</table>

g. Click OK in the Create Part dialog.

h. Select OrderProcessorFault and then select part in the right side of WSDL component palette.

The Create Part dialog displays.

i. Enter and select the following values:
Creating Catch Branches for the Scope_AuthorizeCreditCard

Creating the First Half of the OrderProcessor BPEL Process

5-27

j. Click OK in the Create Part dialog.

k. Select OrderProcessorFault and then select part in the right side WSDL component palette.

The Create Part dialog displays.

l. Enter and select the following values:

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Part Name</td>
<td>orderId</td>
</tr>
<tr>
<td>Reference Type</td>
<td>type</td>
</tr>
<tr>
<td>Part Name</td>
<td>xsd:string</td>
</tr>
</tbody>
</table>

m. Click OK in the Create Part dialog.

The Messages section displays the parts for the OrderProcessorFault message.

5. Create the processFault operation for OrderProcessorCallback:

a. In the Port Types section, select OrderProcessorCallback and then select operation in the right side WSDL component palette.

The Create Operation dialog displays.

b. Enter and select the following values:

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operation Name</td>
<td>processFault</td>
</tr>
<tr>
<td>Operation Type</td>
<td>Request Response</td>
</tr>
<tr>
<td>Input</td>
<td>client:OrderProcessorFault</td>
</tr>
<tr>
<td>Output</td>
<td>client:OrderProcessorResponseMessage</td>
</tr>
</tbody>
</table>

c. Click OK in the Create Operation dialog.

d. In the Port Types section, expand OrderProcessorCallback and processFault.

e. Right-click the unneeded output in processFault and select Delete.
Creating Catch Branches for the Scope_AuthorizeCreditCard

5.10.2 Task 2: Create the gOrderProcessorFaultVariable Variable

You now create the \texttt{gOrderProcessorFaultVariable} variable as input for the branches in the next task.

To create this variable:

1. In the workspace for the \texttt{OrderProcessor} BPEL process, click the \texttt{Variables} icon.
   The Variables dialog displays.
2. Click the \texttt{Add} icon to add a variable.

   The Create Variable dialog displays.
3. In the \texttt{Name} field, enter \texttt{gOrderProcessorFaultVariable}.
4. In the \texttt{Type} section, select \texttt{Message Type} and then select the \texttt{Browse} icon to the right of the \texttt{Message Type} field.
   The Type Chooser dialog appears with a list of available services.
5. Expand \texttt{Message Types} > \texttt{Project WSDL Files} > \texttt{OrderProcessor.wsdl} > \texttt{Message Types} and select \texttt{OrderProcessorFault}, which you added to the WSDL file in Section 5.10.2, "Task 2: Create the gOrderProcessorFaultVariable Variable."

6. Select \texttt{Save All} from the \texttt{File} main menu to save your work.
7. Click X in the \texttt{OrderProcessor.wsdl} tab to close the WSDL Editor.
Creating Catch Branches for the Scope_AuthorizeCreditCard

5.10.3 Task 3: Add Catch Branches to the Scope_AuthorizeCreditCard

To add catches to the Scope_AuthorizeCreditCard scope:

1. Click the Add Catch Branch icon for the scope, as shown in the following figure:

2. Double-click the catch to display the Catch dialog.

3. In the Fault QName section, click the Browse icon.
4. In the Fault Chooser dialog, expand **System Faults** and select **selectionFailure**, and then click **OK**. This catch provides a built-in system fault. It is raised from the **Assign_CreditCheckInput** activity if any of the fields are blank, such as no account number.

5. Click **OK** in the Catch dialog.

6. Click the **Expand (+)** icon to expand the **selectionFailure** catch.

7. In the **selectionFailure** catch, create an assign activity to assign expression 'CreditCardCheck - NO CreditCard' as input to global variable **gOrderProcessorFaultVariable** for orders without credit card numbers.
   a. From the Component Palette, drag an **Assign** activity into the branch.
   b. Rename this activity by double-clicking name underneath the icon. Do not double-click the assign icon itself.
   c. In the edit field, enter **Assign_noCCNumber**.
   d. Double-click **Assign_noCCNumber**.

   The Assign dialog displays.

   e. From the dropdown list, select **Copy Operation**.

   The Create Copy Operation dialog appears.

   f. Enter and select the following values:

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>From</td>
<td>Type Expression</td>
</tr>
<tr>
<td>Expression</td>
<td>string('CreditCardCheck - NO CreditCard')</td>
</tr>
<tr>
<td>To</td>
<td>Type Variable</td>
</tr>
<tr>
<td>Variable</td>
<td>Expand Variables &gt; gOrderProcessorFaultVariable and select code. You created this variable in Section 5.10.2, &quot;Task 2: Create the gOrderProcessorFaultVariable Variable.&quot;</td>
</tr>
</tbody>
</table>

   g. Click **OK** to close the Create Copy Operation dialog and return to the Assign dialog.

   The Copy Operation tab in the Assign dialog updates to show the copy operation.

   h. In the Assign dialog, click **OK**.

8. In the **selectionFailure** catch, insert a throw activity after assign activity **Assign_noCCNumber**, so the Scope_AuthorizeCreditCard scope throws a fault:

   a. From the Component Palette, drag a **Throw** activity below **Assign_noCCNumber**.

   b. Double-click the new throw activity.

   The Throw activity dialog displays.

   c. Enter the following values:
d. Click the Browse Fault Variables icon next to the Fault Variable field.

e. In the Variable Chooser dialog, select the gOrderProcessFaultVariable and click OK.

f. Back in the Throw dialog, click OK.

9. Click the Collapse (-) icon to minimize the catch.

10. Click the Add Catch Branch icon for the scope to create a second catch.

11. Double-click the new catch to display the Catch dialog.

12. Enter the following values:

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Namespace</td>
<td><a href="http://www.globalcompany.example.com/ns/OrderProcessor">http://www.globalcompany.example.com/ns/OrderProcessor</a></td>
</tr>
<tr>
<td>Local Part</td>
<td>OrderProcessorFault</td>
</tr>
</tbody>
</table>

13. Click OK in the Catch dialog.

14. Click the Expand (+) icon to expand the InvalidCredit catch.

15. In the InvalidCredit catch, assign data to take the credit card type and assign it to global variable gOrderProcessorFaultVariable for orders without a valid credit card type.

a. From the Component Palette, drag an Assign activity into the branch.

b. Rename this activity by double-clicking name underneath the icon. Do not double-click the assign icon itself.

c. In the edit field, enter Assign_InvalidCreditFault.

d. Double-click Assign_InvalidCreditFault.

   The Assign dialog displays.

e. From the dropdown list, select Copy Operation.

   The Create Copy Operation dialog appears.

f. Enter and select the following values:

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>From</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Expression</td>
</tr>
</tbody>
</table>
Creating Catch Branches for the Scope_AuthorizeCreditCard

1. Select the XPath Expression Builder icon. The Expression Builder displays.

2. In the BPEL Variables section, expand Variables > gOrderInfoVariable > ns8:orderInfoVOSDO and select CardTypeCode.

3. Click Insert Into Expression.
   The Expression box updates with the following expression:
   bpws:getVariableData('gOrderInfoVariable', '/ns2:orderInfoVOSDO/ns8:CardTypeCode')

4. Prepend the expression with the following Expression box, enter the following:
   concat(

5. Append the expression with the following Expression box, enter the following:
   , ' is not a valid creditcard type')
   The expression should now looks like the following:
   concat(bpws:getVariableData('gOrderInfoVariable', '/ns4:orderInfoVOSDO/ns4:CardTypeCode'), ' is not a valid creditcard type')

6. Click OK to close the Expression Builder.

To
- Type Variable
- Variable Expand Variables > gOrderProcessorFaultVariable and select summary.

Click OK to close the Create Copy Operation dialog and return to the Assign dialog.

The Copy Operation tab in the Assign dialog updates to show the copy operation.

16. In the InvalidCredit catch, assign data expression 'CreditCardCheck - NOT VALID' to global variable gOrderProcessorFaultVariable.

a. From the dropdown list in the Assign dialog, select Copy Operation.
   The Create Copy Operation dialog appears.

b. Enter and select the following values:

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>From</td>
<td></td>
</tr>
<tr>
<td>- Type</td>
<td>Expression</td>
</tr>
</tbody>
</table>
c. Click OK to close the Create Copy Operation dialog and return to the Assign dialog. The Copy Operation tab in the Assign dialog updates to show the copy operation.

d. In the Assign dialog, click OK.

17. In the InvalidCredit catch, insert a throw activity after assign activity Assign_InvalidCreditFault, so the Scope_AuthorizeCreditCard scope throws a fault:

   a. From the Component Palette, drag a Throw activity below Assign_InvalidCreditFault.

   b. Double-click the new throw activity. The Throw activity dialog displays.

   c. Enter the following values:

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expression</td>
<td>string('CreditCardCheck - NOT VALID')</td>
</tr>
<tr>
<td>Type</td>
<td>Variable</td>
</tr>
<tr>
<td>Variable</td>
<td>Expand Variables &gt; gOrderProcessFaultVariable and select code.</td>
</tr>
</tbody>
</table>

   d. Click the Browse Fault Variables icon next to the Fault Variable field.

   e. In the Variable Chooser dialog, select the gOrderProcessorFaultVariable and click OK.

   f. Click OK in the Throw dialog.

18. Click the Collapse (-) icon to minimize the catch.

19. Click the Collapse (-) icon to minimize the Scope_AuthorizeCreditCard scope.

20. Select Save All from the File main menu to save your work.

5.11 Creating the RequiresApprovalRule Business Rule

You create a business rule activity to specify the RequiresApprovalRule business rule. This rule specifies that if the order total is $2,000 or more, then a manager’s approval is required.

Another activity uses the output from the business rule to either automatically approve the order or use a human task to obtain manager approval.
When you add a business rule activity to a BPEL process, you can create input and output variables to provide input to the business rule activity, and to obtain results from the business rule activity.

To use business rules with Oracle JDeveloper, you perform the following:

- Add a business rule activity to the BPEL process
- Create input and output variables in the BPEL process
- Create an Oracle Business Rules dictionary in the project

If you want, you can associate a business rule service component created in the SOA Composite Editor with a BPEL process service component. You create this association with the business rule activity of the BPEL process. This activity creates a business rule partner link. This activity also enables you to create copy operation assignments between the fact data in your rule set and BPEL variables. When complete, a business rule activity is created that consists of assign and invoke activities to the business rule partner link.

To create the RequiresApprovalRule business rule, perform the following tasks:

- Section 5.11.1, "Task 1: Create Scope_CheckApprovalLimit Scope"
- Section 5.11.2, "Task 2: Add the IOrderApproved Variable"
- Section 5.11.3, "Task 3: Create the Assign_DefaultNotRequiresApproval Assign Activity"
- Section 5.11.4, "Task 4: Create the RequiresApprovalRule Business Rule"
- Section 5.11.5, "Task 5: Reference the RequiresApprovalRule Dictionary in the BPEL Designer"
- Section 5.11.6, "Task 6: Define a Variable in Rules Designer"
- Section 5.11.7, "Task 7: Add a New Rule for the Ruleset in Rules Designer"
- Section 5.11.8, "Task 8: Redeploy the OrderBookingComposite Composite"
- Section 5.11.9, "Task 9: Initiate a Test Instance for the OrderBookingComposite Composite"

5.11.1 Task 1: Create Scope_CheckApprovalLimit Scope

The Scope_CheckApprovalLimit scope invokes the RequiresApprovalRule business rule, as shown in Figure 5–5.
To add the scope:

1. From the Component Palette, drag a **Scope** activity below the **Scope_AuthorizeCreditCard** scope.
2. Rename this activity by double-clicking the name underneath the icon. Do not double-click the activity icon itself.
3. In the edit field, change the name to **Scope_CheckApprovalLimit**.
4. Click the **Expand (+)** icon to expand the **Scope_CheckApprovalLimit** scope.

### 5.11.2 Task 2: Add the lOrderApproved Variable

In this task, you create local variable **lOrderApproved** variable, which provides input to a business rule variable.

To create this variable:

1. In the workspace for the **Scope_CheckApprovalLimit** scope, click the **Variables** icon.
   
   The Variables dialog displays.
2. Click the **Add** icon to add a variable.
   
   The Create Variable dialog displays.
3. In the **Name** field, enter **lOrderApproved**.
4. In the **Type** section, select **Simple Type** and then select the **Browse XML Schema Types** icon to the right of the field.
   
   The Type Chooser dialog appears with a list of available services.
5. Select **string** under **XML Schema Simple Types**.
6. Click **OK** in the Type Chooser dialog.
7. Click **OK** in the Create Variable dialog.
   
The Variables dialog updates with the `OrderApproved` variable.
8. Click **OK** in the Variables dialog.

### 5.11.3 Task 3: Create the **Assign_DefaultNotRequiresApproval** Assign Activity

In this task, you create an assign activity to take the order total and send it to the input variable for the business rule.

To assign an input variable for the purchase price to the business rule:

1. From the Component Palette, drag an **Assign** activity into the `Scope_CheckApprovalLimit` scope.
2. Rename this activity by double-clicking the name underneath the icon. Do not double-click the assign icon itself.
3. In the edit field, enter `Assign_DefaultNotRequiresApproval`.
4. Double-click the **Assign_DefaultNotRequiresApproval** activity.
   
The Assign dialog displays.
5. From the dropdown list, select **Copy Operation**:
   
The Create Copy Operation dialog appears.
6. Enter and select the following values:

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>From</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td><strong>Expression</strong></td>
</tr>
<tr>
<td>Expression</td>
<td><code>string('false')</code></td>
</tr>
</tbody>
</table>
Creating the RequiresApprovalRule Business Rule

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>To</td>
<td></td>
</tr>
<tr>
<td>■ Type</td>
<td>Variable</td>
</tr>
<tr>
<td>■ Variable</td>
<td>Expand Scope - Scope_CheckApprovalLimit &gt; Variables and select IOrderApproved, which is the variable you created in Section 5.11.2, &quot;Task 2: Add the IOrderApproved Variable.&quot;</td>
</tr>
</tbody>
</table>

7. Click OK to close the Create Copy Operation dialog and return to the Assign dialog.
   The Copy Operation tab in the Assign dialog updates to show the copy operation.
8. In the Assign dialog, click OK.
9. Select Save All from the File main menu to save your work.

5.11.4 Task 4: Create the RequiresApprovalRule Business Rule

To create the business rule:
1. Click the composite.xml tab to view the SOA Composite Editor.
2. Drag a Business Rule service component into the SOA Composite Editor.
   The Create Business Rule dialog displays.
3. In the Name field, enter RequiresApprovalRule to be the name of the Oracle Business Rules dictionary.
4. Leave the default for the Package field.
5. In the Inputs/Outputs section, from the Add menu, select Input to select the input for the business rule:
   The Type Chooser dialog displays.
6. Import the complete schema located in the DEMO_DOWNLOAD_HOME directory:
   a. Click the Import Schema File icon.
      The Import Schema File dialog displays.
   b. Select File System and in the Location section, browse for OrderBookingRules.xsd in DEMO_DOWNLOAD_HOME/CompositeServices/OrderBookingComposite/xsd and click OK.
   c. In the Import Schema dialog, ensure the OrderBookingRules.xsd now displays in the URL field and the Copy to Project option is selected, and then click OK.
      The Localized Files dialog displays, prompting you to import the OrderBookingRules.xsd schema file.
   d. Deselect the Maintain original directory for imported files option and click OK to import the file.
      The Type Chooser dialog displays.
7. Select the input for the business rule:
a. In the Type Chooser dialog, expand OrderBookingRules.xsd and select approve.
b. Click OK to return to the Create Business Rules dialog.

8. In the Inputs/Outputs section, select the output for the business rule:
   a. From the Add menu, select Output.
      The Type Chooser dialog displays.
   b. Expand OrderBookingRules.xsd and select approve.
   c. Click OK to return to the Create Business Rules dialog.

9. Click OK to create the business rule.
   The RequiresApprovalRule business rule displays in the composite.

10. Select Save All from the File main menu to save your work.

11. Double-click RequiresApprovalRule in the SOA Composite Editor.
    Oracle JDeveloper displays the Rules Designer, with the dictionary in the RequiresApprovalRule.rules tab.

    For an overview of the Rules Designer, see Oracle Fusion Middleware Developer’s Guide for Oracle SOA Suite.
    As part of the Business Rule designer, a new rule dictionary is created with the following pre-loaded data:
    - XML fact type model based on the input and output metadata information of the business rule activity.
    - A new ruleset that must be completed by adding rules to it.
    - A new service with the input and output contract of the business rule activity. The service is being invoked from the activity at run time.
    - A new business rule service component for the rule dictionary. You wire it to the BPEL process in Section 5.11.5, “Task 5: Reference the RequiresApprovalRule Dictionary in the BPEL Designer.”
    You modify this rule in a later task. Therefore, do not close the tab.

5.11.5 Task 5: Reference the RequiresApprovalRule Dictionary in the BPEL Designer

    To reference the RequiresApprovalRule in the Scope_CheckApprovalLimit scope:
    1. Click the OrderProcessor.bpel tab.
2. From the Component Palette, drag a Business Rule activity below the Assign_DefaultNotRequiresApproval activity in the Scope_CheckApprovalLimit scope. The Business Rule dialog displays.

3. In the Name field, enter BusinessRule_ApprovalRequired.

4. From the Dictionary list, select RequiresApprovalRule, which is the rule you created in Section 5.11.4, "Task 4: Create the RequiresApprovalRule Business Rule."

5. Leave the default settings for the Service and Operation fields.

6. Create input to the business rule, so that the gOrderInfoVariable and lOrderApproved variables assign data to input variable com_example_globalcompany_ns_orderbookingservice_rules_Approve_i for the business rule.

   a. In the Assign Input Facts tab, click the Create icon. The Decision Fact Map dialog displays.

   b. Enter and select the following values:

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>From</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Variable</td>
</tr>
<tr>
<td>Variable</td>
<td>Expand Variables &gt; gOrderInfoVariable &gt; ns4:orderInfoVOSDO and select ns4:OrderTotal. Note: The namespace number values (for example, ns1, ns2) can vary.</td>
</tr>
</tbody>
</table>

    c. Click OK. The copy operation displays in the Assign Input Facts tab of the Business Rule dialog.

    d. In the Assign Input Facts tab, click the Create icon again to create a second assignment. The Decision Fact Map dialog displays.

    e. Enter and select the following values:

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>From</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Variable</td>
</tr>
<tr>
<td>Variable</td>
<td>Expand Scope &gt; Variables and select lOrderApproved. Note: The namespace number values (for example, ns1, ns2) can vary.</td>
</tr>
</tbody>
</table>

| From      |       |
| Type      | Variable |
| Variable  | Expand com_example_globalcompany_ns_orderbookingservice_rules_Approve_i > ns10:approve and select ns10:price. |

Creating the RequiresApprovalRule Business Rule
Creating the RequiresApprovalRule Business Rule

5.11.6 Task 6: Define a Variable in Rules Designer

You define variables in the data model. When you make changes later, you must edit the value of the variable. You create a variable named MAX_PRICE to define the dollar amount where orders above this amount would need manual approval from a manager and orders under this amount are approved automatically. You set the MAX_PRICE variable to $2000.

To create the MAX_PRICE variable:
1. In the Rules Designer, select the Globals tab.

---

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Business Rule Facts</td>
</tr>
<tr>
<td>Variable</td>
<td>Expand com_example_globalcompany_ns_orderbookingservice_rules_Approve_i &gt; ns10:approve and select ns10:approvalRequired.</td>
</tr>
</tbody>
</table>

f. Click OK.

The two copy operations display in the Assign Input Facts tab of the Business Rule dialog.

7. Create output from the business rule, so that the variable com_example_globalcompany_ns_orderbookingservice_rules_Approve_o for the business rule sends data to the IOrderApproved variable for the scope:

a. Click the Assign Output Facts tab.

b. In the Assign Output Facts tab, click the Create icon.

The Decision Fact Map dialog displays.

c. Enter and select the following values:

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>From</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Business Rule Facts</td>
</tr>
<tr>
<td>Variable</td>
<td>Expand com_example_globalcompany_ns_orderbookingservice_rules_Approve_o &gt; ns10:approve and select ns10:approvalRequired.</td>
</tr>
<tr>
<td>Note: The namespace number values (for example, ns1, ns2) can vary.</td>
<td></td>
</tr>
</tbody>
</table>

| To | |
| Type | Variable |
| Variable | Expand Scope - Scope_CheckApprovalLimit > Variables and select IOrderApproved. |

d. Click OK.

The copy operation displays in the Assign Input Facts tab of the Business Rule dialog.


9. Select Save All from the File main menu to save your work.
2. Click the Create icon to add a new variable entry.
   The Edit Variable dialog displays.
3. In the Name field, enter MAX_PRICE.
4. In the Description field, enter the following string:
   Limit for Automatic Approval
5. From Type dropdown list, select int.
6. Do not select any value for Bucketset.
7. Click the icon next to the Value field.
   The Expression Builder dialog displays.
8. In the Expression area, enter 2000, and then click OK to close the Expression Builder.
9. In the Edit Variable dialog, click OK.

5.11.7 Task 7: Add a New Rule for the Ruleset in Rules Designer
When you create the rule dictionary, an empty ruleset named Ruleset_1 was created without any rules. In this task, you add the CheckOrderTotalAgainstLimit rule. This rule specifies that if the order is greater than or equal to $2,000, then the order requires manual approval.

To create the rule:
1. In Rules Designer, select Ruleset_1 from the left menu.
2. From the Create dropdown list, select Create Rule.
   A new rule displays:
   3. Click <insert test> to display the statement template.
   4. In the IF section, click the left-hand operand and select approve.price.
5. Click the operator and select \( \geq \).

6. Click the right-hand operand and select MAX_PRICE.

7. In the THEN section, click \(<\text{insert action}>\) and select modify.

8. Click \(<\text{target}>\) and select approve.

9. Click \(<\text{add property}>\).

The Properties dialog displays.

10. In the approvalRequired row, select true from the Value dropdown menu and click the Constant check box.

11. Click Close.

The business rule updates and is complete.

12. Select Save All from the File main menu to save your work.

13. Click X in the RequiresApprovalRule.rules tab to close the Rules Designer.

**5.11.8 Task 8: Redeploy the OrderBookingComposite Composite**

To redeploy the OrderBookingComposite composite:

1. In the Application Navigator, right-click OrderBookingComposite and select Deploy > OrderBookingComposite > to MyAppServerConnection.

The SOA Deployment Configuration Dialog displays.

2. Select Overwrite any existing composite with the same revision ID to overwrite the composite you deployed in Section 5.7.4, "Task 4: Deploy the OrderBookingComposite Composite."
3. When prompted with the Authorization Request dialog, enter weblogic in the Username field and the password in the Password field.

In SOA - Log, a series of validations display, followed by:

BUILD SUCCESSFUL
Total time: nn seconds

5.11.9 Task 9: Initiate a Test Instance for the OrderBookingComposite Composite

Initiate a test instance of the OrderBookingComposite composite, as you did in Section 5.7.6, "Task 6: Initiate a Test Instance for the OrderBookingComposite Composite." This time, in the Flow Trace window, notice how the order progresses through the Scope_CheckApprovalLimit scope.

5.12 Adding the Switch_ApprovalRequired Switch to the Scope_CheckApprovalLimit Scope

For an order that requires manual approval because the order total is $2,000 or more, the SwitchApprovalRequired switch in the ScopeCheckApprovalLimit scope consists of a <case> branch that passes control to the ApprovalHumanTask human task activity, which enables a manager named jstein to approve or reject the orders. For orders that do not require manual approval, this switch does not apply to them. Figure 5–6 shows the SwitchApprovalRequired switch contains a human task activity and another switch activity to handle the manager’s response.

Figure 5–6 Switch_ApprovalRequired Switch with Human Task and Switch

To create the SwitchApprovalRequired switch, perform the following tasks:

- Task 1: Create the Switch_ApprovalRequired Switch
5.12.1 Task 1: Create the Switch_ApprovalRequired Switch

To create the Switch_ApprovalRequired switch:

1. Click the OrderProcessor.bpel tab.
2. From the Component Palette, drag a Switch activity from the Component Palette section to below the BusinessRule_ApprovalRequired activity.
3. Rename this activity by double-clicking the name underneath the icon. Do not double-click the activity icon itself.
4. In the edit field, change the name to Switch_ApprovalRequired.
5. Click the Expand (+) icon to expand the Switch_ApprovalRequired scope.
6. Remove the unneeded <otherwise> branch.
   a. Right-click the <otherwise> branch and select Delete from the menu.
   b. When prompted to remove the branch, click Yes.

   The switch looks similar to the one in the following figure:

   ![Switch_ApprovalRequired Switch](image)

7. Select Save All from the File main menu to save your work.

5.12.2 Task 2: Set the Condition for the <case> Branch

To create the <case> branch:

1. Double-click the title bar of the <case> box to display the Switch Case dialog. The Switch Case dialog displays.
2. In the Name box, enter:
   approval required = true
3. Click the XPath Expression Builder icon above the Expression box to display the Expression Builder dialog.
4. In the BPEL Variables box, expand Scope - Scope_CheckApprovalLimit > Variables and select lOrderApproved.

lOrderApproved is the variable you defined in Section 5.11.2, "Task 2: Add the lOrderApproved Variable."

The Content Preview box should show the following:

bpws:getVariableData('lOrderApproved')

5. Click Insert Into Expression. The Expression box displays the function with the one parameter.

6. Append = 'true' to the expression, starting with a space, in the Expression box so that the expression looks like this:

bpws:getVariableData('lOrderApproved') = 'true'

7. Click OK in the Expression Builder dialog. The Switch Case dialog now contains the expression.

8. Click OK in the Switch Case dialog to close it.

9. Select Save All from the File main menu to save your work.

You modify the <case> branch with two activities, a human task activity and a switch activity). To create these activities, you must create a sequence activity to be the container for these two activities.

5.12.3 Task 3: Create a Human Task in the <case> Branch to Approve an Order

To create a human task in the <case> branch:

1. Drag a Human Task activity into the <case> box.

The Create a Human Task dialog appears.

2. From the Task Definition list, click the Create icon next to the Task Definition field.

The Create Human Task dialog displays.

3. Enter and select the following values:

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>ApprovalHumanTask</td>
</tr>
<tr>
<td>Priority</td>
<td>Leave the priority set to 3 (Normal).</td>
</tr>
<tr>
<td>Namespace</td>
<td><a href="http://www.globalcompany.com/ns/OrderBooking/ApprovalHumanTask">http://www.globalcompany.com/ns/OrderBooking/ApprovalHumanTask</a></td>
</tr>
<tr>
<td>Title</td>
<td>Do not enter a value for the time being.</td>
</tr>
</tbody>
</table>

4. In the Parameters section, select the Add Task Parameter icon.

The Add Task Parameter dialog displays.

5. Click the icon next to the Source field to launch the Task Parameters page for selecting a parameter source.

6. Expand Variables > gOrderInfoVariable > ns8:orderInfoVOSDO and select the OrderId key.

7. Click OK to close the Task Parameters dialog.
8. Back in the Add Task Parameter dialog, in the **Parameter Name** field, modify the field to `orderId`.

9. Click **OK** to close the Add Task Parameter dialog and return to the Create Human Task dialog.

10. In the Create Human Task dialog, click **OK**.

    The Human Task Editor appears.

    For an overview of the Human Task Editor, see *Oracle Fusion Middleware Developer’s Guide for Oracle SOA Suite*.

11. Define a title for the task
a. In the **Title** field, enter:

```
Approval Required for Order Id:
```

b. Click the icon to the right of the **Title** field to select the order ID. The Expression Builder dialog displays.

c. From the **Schema** section, expand `task:task > task:payload` and select `task:orderId`.

d. Click **Insert Into Expression**.

The Expression box updates with the following expression:

```
/task:task/task:payload/task:orderId
```

e. Click **OK** to close the Expression Builder.

The ApprovalHumanTask.task pages updates with the following for the title:

```
Approval Required for Order Id:<%/task:task/task:payload/task:orderId%>>
```

12. Specify an approver for the human task:

a. In the **Assignment and Routing Policy** section, double-click `<no participants>`. 
Adding the Switch_ApprovalRequired Switch to the Scope_CheckApprovalLimit Scope

The Add Participant Type dialog displays.

b. In the Label field, enter Approver.

c. From the dropdown menu in the Participant Names table, select Add User.

A new row displays in the Participant Names table.

d. Click the Browse icon to the far right of the row.

The Identity Lookup dialog appears.

e. From the Application Server list, select MyAppServerConnection.

f. Click the Lookup icon. It is located to the right of the User Name field.

The search results display in the Search User section.

g. Select jstein and then click Select.

jstein is added to the Selected User section.

h. Click OK to close the Identity Lookup dialog.

jstein displays in the Participants Names table in the Edit Participant Type dialog.

i. Click OK in the Add Participant Type dialog.

Approver displays in the Assignment and Routing Policy section of the ApprovalHumanTask.task window.
13. Select **Save All** from the **File** main menu to save your work.

14. Click **X** in the **ApprovalHumanTask.task** tab to close the human task.

15. Click the **composite.xml** tab to see the human task. **ApprovalHumanTask** displays in the SOA Composite Editor.

5.12.4 Task 4: Modify TaskSwitch Activity in <case> Branch to Handle Manager’s Response

Oracle JDeveloper created a switch called **taskSwitch** for you automatically after the human task activity in the **OrderProcessor** BPEL process, as shown in Figure 5–7.

*Figure 5–7  taskSwitch Activity in the <case> Branch*
Adding the Switch_ApprovalRequired Switch to the Scope_CheckApprovalLimit Scope

This switch enables you to define the actions to take depending on whether the manager approved or rejected the order, or if the order has expired.

The switch handles these cases:

- The manager rejected the order.
- The manager approved the order.
- The order has expired.

To modify the switch for these actions:

1. Click the Expand (+) icon to expand the taskSwitch.

2. In the <case Task outcome is REJECT> branch, remove the CopyPayloadFromTask activity and replace it with a throw activity.

   For those orders not approved, this activity throws a fault called Throw_OrderProcessorFault. The OrderProcessor process terminates after executing the throw activity.

   a. Right-click the CopyPayloadFromTask activity and select Delete from the menu.
   b. When prompted to remove the branch, click Yes.
   c. From the Component Palette, drag a Throw activity into the <case Task outcome is REJECT> branch.
   d. Double-click the new throw activity.

   The Throw activity dialog displays.

   e. Enter and select the following values:

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Throw_OrderProcessorFault</td>
</tr>
<tr>
<td>Namespace</td>
<td><a href="http://www.globalcompany.example.com/ns/OrderBookingService">http://www.globalcompany.example.com/ns/OrderBookingService</a></td>
</tr>
<tr>
<td>Local Part</td>
<td>OrderProcessorFault</td>
</tr>
</tbody>
</table>
   | Fault Variable| 1. Click the Browse icon.  
                | 2. In the Variable Chooser dialog, select gOrderProcessorFaultVariable.  
                | 3. Click OK.                                        |

   f. Click OK.

3. In the <case Task outcome is APPROVE> branch, remove the CopyPayloadFromTask activity and replace it with an empty assign activity. the empty assign passed the output from the business to the next scope in the process.

   a. Right-click the CopyPayloadFromTask activity and select Delete from the menu.
   b. When prompted to remove the branch, click Yes.
   c. From the Component Palette, drag an Empty activity into the <case Task outcome is APPROVE> branch.
   d. Rename this activity by double-clicking the name underneath the icon.

   e. In the edit field, remove the name.
4. In the Assign dialog, click OK.

5. In the <otherwise> branch, remove the CopyPayloadFromTask activity and replace it with a throw activity.

   For those orders not approved, this activity throw a fault called ThrowRejected. The OrderProcessor process terminates after executing the throw activity.
   a. Right-click the CopyPayloadFromTask activity and select Delete from the menu.
   b. When prompted to remove the branch, click Yes.
   c. From the Component Palette, drag a Throw activity into the <otherwise Task> branch.
   d. Double-click the new throw activity.
      The Throw activity dialog displays.
   e. Enter and select the following values:

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Throw_OrderProcessingFault</td>
</tr>
<tr>
<td>Namespace</td>
<td><a href="http://www.globalcompany.example.com/ns/OrderBookingService">http://www.globalcompany.example.com/ns/OrderBookingService</a></td>
</tr>
<tr>
<td>Local Part</td>
<td>OrderProcessorFault</td>
</tr>
<tr>
<td>Fault Variable</td>
<td>1. Click the Browse icon.</td>
</tr>
<tr>
<td></td>
<td>2. In the Variable Chooser dialog, select gOrderProcessorFaultVariable.</td>
</tr>
<tr>
<td></td>
<td>3. Click OK.</td>
</tr>
</tbody>
</table>

f. Click OK.

6. Click the Collapse (-) icon to minimize taskSwitch switch.

7. Click the Collapse (-) icon to minimize the Scope_CheckApprovalLimit scope.

8. Select Save All from the File main menu to save your work.

5.12.5 Task 5: Redeploy the OrderBookingComposite Composite

To redeploy the OrderBookingComposite composite:

1. In the Application Navigator, right-click OrderBookingComposite and select Deploy > OrderBookingComposite > to MyAppServerConnection.
   The SOA Deployment Configuration Dialog displays.

2. Select Overwrite any existing composite with the same revision ID to overwrite the composite you previously deployed.

3. When prompted with the Authorization Request dialog, enter weblogic in the Username field and the password in the Password field.
   In SOA - Log, a series of validations display, followed by:
   BUILD SUCCESSFUL
   Total time: nn seconds
5.12.6 Task 6: Initiate a Test Instance for the OrderBookingComposite Composite

Initiate a test instance of the OrderBookingComposite composite, as you have done previously. See Section 5.7.6, "Task 6: Initiate a Test Instance for the OrderBookingComposite Composite" for further information on creating a test instance. For this test instance, make the following adjustments:

■ In the Inputs Arguments section, in the orderID field, enter an ID over 1000 to submit an order for over $2000.

■ In the Flow trace window, scroll toward the bottom of the Scope_CheckApprovalLimit scope to see the ApprovalHumanTask human task was initiated after the business rule.

Proceed with Chapter 6, "Creating the Second Half of the OrderProcessor BPEL Process," to complete the creation of the OrderProcessor BPEL process. After the process is complete, you can create a form for the manager to approve orders in Chapter 9, "Creating the Task Display Form for the ApprovalHumanTask Human Task."
Creating the Second Half of the OrderProcessor BPEL Process

This chapter describes how to create the second half of the OrderProcessor BPEL process for composite OrderBookingComposite. This chapter assumes you have performed all the tasks from Chapter 3 to Chapter 5.

This chapter contains the following sections:

■ Section 6.1, "Overview of Tasks for Creating the Second Half of OrderProcessor"
■ Section 6.2, "Creating the Scope_RetrieveQuotes Flow"
■ Section 6.3, "Creating the Scope_SelectPreferredSupplier Scope"
■ Section 6.4, "Creating the Services and Routing Required for the Scope_FulfillOrder Scope"
■ Section 6.5, "Creating the Scope_FulfillOrder Scope"
■ Section 6.6, "Creating the Scope_UpdateStatusToComplete Scope for Completed Orders"
■ Section 6.7, "Creating the Scope_NotifyCustomerofCompletion Scope"
■ Section 6.8, "Adding a Catch Branch for Incomplete Orders for the Entire Process"

6.1 Overview of Tasks for Creating the Second Half of OrderProcessor

Table 6–1 lists and describes the tasks for creating the second half of the OrderProcessor BPEL process for the OrderBookingComposite composite.
6.2 Creating the Scope_RetrieveQuotes Flow

The Scope_RetrieveQuotes flow sends the order information to two suppliers, an internal warehouse and an external partner warehouse, and the warehouses return their bids for the orders. The Scope_SelectPreferredSupplier scope then chooses the warehouse based on the lowest bid.

Figure 6–1 shows the activities in the Scope_RetrieveQuotes scope. The scope uses a flow activity to send the order information to the two warehouses in parallel.
Figure 6–1  Activities for the Scope_RetrieveQuotes Scope

Figure 6–2 shows the OrderProcessor BPEL process invokes the InternalWarehouse service and receives a quote from it. This service references a WSDL file from a BPEL process also named InternalWarehouseService, which resides within the OrderBookingComposite composite. Figure 6–3 shows how it appears in the SOA Composite Editor. This supplier statically returns a value of $1,000 for all orders regardless of their order total.
Figure 6–2  Activities for Receiving a Bid from an Internal Warehouse

Figure 6–3  InternalWarehouseService in OrderBookingComposite

Figure 6–4  show the OrderProcessor BPEL process invokes the PartnerSupplierMediator mediator and receives a quote from it. This mediator initiates the ExternalPartnerSupplier BPEL process in the PartnerSupplierComposite, which you created in Chapter 1, "Introduction to the SOA Sample Application." Figure 6–5 shows how the mediator and service appear in the SOA Composite Editor.
To create the Scope_RetrieveQuotes scope and related services, perform the following tasks:

- Task 1: Add the Scope_RetrieveQuotes Scope
- Task 2: Create the InternalWarehouseService BPEL Process
- Task 3: Modify the InternalWarehouseService Process
- Task 4: Wire OrderProcessor to the InternalWarehouseService Process
- Task 5: Create the PartnerSupplierService Service
Creating the Scope_RetrieveQuotes Flow

- Task 6: Create a PartnerSupplier Mediator Service for the PartnerSupplierService
- Task 7: Create Routing Rules Between the PartnerSupplierMediator Mediator to the ExternalPartnerSupplier Service
- Task 8: Wire OrderProcessor to the PartnerSupplierMediator Mediator
- Task 9: Add the gWarehouseQuotes Variable
- Task 10: Add Activities to Obtain a Quote from the InternalWarehouse Process
- Task 11: Add Activities to Obtain a Quote from the PartnerSupplierMediator Mediator

6.2.1 Task 1: Add the Scope_RetrieveQuotes Scope

To add the Scope_RetrieveQuotes scope:

1. From the Component Palette, drag a Scope activity below the Scope_CheckApprovalLimit scope.
2. Rename this activity by double-clicking the name underneath the icon. Do not double-click the activity icon itself.
3. In the edit field, change the name to Scope_RetrieveQuotes.
4. Click the Expand (+) icon to expand the Scope_RetrieveQuotes scope.
5. Drag and drop a Flow activity into the scope.
6. Rename this activity by double-clicking name underneath the icon. Do not double-click the assign icon itself.
7. In the edit field, enter RetrieveQuotesFromSuppliers.
8. Click the Expand (+) icon to expand the Scope_RetrieveQuotes scope.

The flow should now contains two empty sequences.

6.2.2 Task 2: Create the InternalWarehouseService BPEL Process

To create the InternalWarehouseService BPEL process:

1. Click the composite.xml tab.
2. Select SOA from the Component Palette.
3. Drag a BPEL Process from the Service Components list into the canvas workspace.

   The Create BPEL Process dialog appears.

4. Enter and select the following values:

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>InternalWarehouseService</td>
</tr>
<tr>
<td>Namespace</td>
<td><a href="http://www.globalcompany.example.com/ns/InternalWarehouse">http://www.globalcompany.example.com/ns/InternalWarehouse</a></td>
</tr>
<tr>
<td>Template</td>
<td>Asynchronous BPEL Process</td>
</tr>
<tr>
<td>Expose as a SOAP service</td>
<td>Deselect this check box.</td>
</tr>
</tbody>
</table>

5. In the Input field, import the complete schema located in the $DEMO_DOWNLOAD_HOME$ directory.
   a. In the Input field, click the Browse Input Elements icon.
      The Type Chooser dialog displays.
   b. Expand Project Schema Files > InternalWarehouse.xsd and select WarehouseRequest and then click OK.
   c. Click OK.

6. In the Output field, import the complete schema located in the $DEMO_DOWNLOAD_HOME$ directory.
   a. In the Input field, click the Browse Input Elements icon.
      The Type Chooser dialog displays.
   b. Expand Project Schema Files > InternalWarehouse.xsd and select WarehouseResponse and then click OK.

7. Click OK.

   The BPEL process displays in the SOA Composite Editor.

8. Double-click InternalWarehouseService BPEL process to display the BPEL Designer.
6.2.3 Task 3: Modify the InternalWarehouseService Process

Next, you create an assign activity to copy input data to the InternalWarehouseService service:

1. Create the assign activity:
   a. From the Component Palette, drag an Assign activity to below the receiveInput receive activity.
   b. Rename this activity by double-clicking the name underneath the icon. Do not double-click the activity icon itself.
   c. In the edit field, change the name to Assign_Defaults.
   d. Double-click the assign activity.
      The Assign dialog displays.

2. Copy string ‘InternalWarehouse’ to the output variable for the InternalWarehouseService service:
   a. From the dropdown list, select Copy Operation:
      The Create Copy Operation dialog appears.
   b. Enter and select the following values:

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>From</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Expression</td>
</tr>
<tr>
<td>Expression</td>
<td>string('InternalWarehouse')</td>
</tr>
<tr>
<td>To</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Variable</td>
</tr>
<tr>
<td>Variable</td>
<td>Expand Variables &gt; outputVariable &gt; payload &gt; client:WarehouseResponse and select client:warehouse.</td>
</tr>
</tbody>
</table>
Creating the Scope_RetrieveQuotes Flow

Creating the Second Half of the OrderProcessor BPEL Process

3. Assign the current date to the output variable for the InternalWarehouseService service:
   a. From the dropdown list, select Copy Operation:
      The Create Copy Operation dialog appears.
   b. Enter and select the following values:

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>From</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Expression</td>
</tr>
<tr>
<td>Expression</td>
<td>xp20:current-date()</td>
</tr>
<tr>
<td>To</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Variable</td>
</tr>
<tr>
<td>Variable</td>
<td>Expand Variables &gt; outputVariable &gt; payload &gt; ns3:WarehouseResponse and select ns3:deliveryDate. The namespace number values (for example, ns1, ns2) can vary.</td>
</tr>
</tbody>
</table>

   c. Click OK to close the Create Copy Operation dialog and return to the Assign dialog.
      The Copy Operation tab in the Assign dialog updates to show the rule.

4. Assign 1000 to the output variable for the InternalWarehouseService service. This value ensures this warehouse statically returns a value of $1,000 for all orders.
   a. From the dropdown list, select Copy Operation:
      The Create Copy Operation dialog appears.
   b. Enter and select the following values:

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>From</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Expression</td>
</tr>
<tr>
<td>Expression</td>
<td>1000</td>
</tr>
<tr>
<td>To</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Variable</td>
</tr>
<tr>
<td>Variable</td>
<td>Expand Variables &gt; outputVariable &gt; payload &gt; ns3:WarehouseResponse and select ns3:orderTotal. The namespace number values (for example, ns1, ns2) can vary.</td>
</tr>
</tbody>
</table>

   c. Click OK to close the Create Copy Operation dialog and return to the Assign dialog.
      The Copy Operation tab in the Assign dialog updates to show the rule.
5. In the Assign dialog, click OK.
6. Select Save All from the File main menu to save your work.
7. Click X in the InternalWarehouseService.bpel tab to close the process.

### 6.2.4 Task 4: Wire OrderProcessor to the InternalWarehouseService Process

To wire the OrderProcessor BPEL process to the InternalWarehouseService service:

1. Drag a wire from OrderProcessor to the InternalWarehouseService reference handle.

You should see a wire going from the OrderProcessor BPEL process to the InternalWarehouseService process in the SOA Composite Editor.

2. Click the OrderProcessor.bpel tab to see the InternalWarehouseService in the Partner Links lane of the BPEL Designer.
3. Select Save All from the File main menu to save your work.

6.2.5 Task 5: Create the PartnerSupplierService Service

The PartnerSupplierMediator mediator obtains a quote from a partner warehouse service named PartnerSupplierService, which calls the PartnerSupplierComposite you created in Chapter 3. Before you create the PartnerSupplierMediator mediator, you must create the PartnerSupplierService service. This service references the WSDL created when you created the ExternalPartnerSupplier BPEL process in Chapter 3.

To create a Web service for the PartnerSupplierService service:

1. If the PartnerSupplierComposite composite was not deployed when you created the composite, deploy it now:
   a. In the Application Navigator, right-click PartnerSupplierComposite and select Deploy > PartnerSupplierComposite > to MyAppServerConnection.
      The SOA Deployment Configuration Dialog displays.
   b. Accept the default settings and click OK.
   c. When prompted with the Authorization Request dialog, enter weblogic in the Username field and the password in the Password field.
      In SOA - Log, a series of validations display, followed by:
      BUILD SUCCESSFUL
      Total time: nn seconds

2. In the composite.xml tab, from the Component Palette, drag a Web Service from the Service Adapters list into the right swim lane of the SOA Composite Editor.
   The Web Service window appears.
3. In the Name field, enter PartnerSupplierService.
4. From the Type list, select Reference.
5. Click the Find existing WSDLs icon next to the WSDL URL field.
   The SOA Resource Lookup dialog displays.
7. Expand Application Server > MyAppServerConnection > SOA > PartnerSupplierComposite and select externalpartnersupplier_client_ep.
8. Click OK in the SOA Resource Lookup dialog.
9. In the Web Service dialog, from the Port Type list, select ExternalPartnerSupplier and from the Callback Port Type list, select ExternalPartnerSupplierCallback.

10. Click OK in the Web Service dialog.

The PartnerSupplierService service displays in the SOA Composite Editor.

6.2.6 Task 6: Create a PartnerSupplier Mediator Service for the PartnerSupplierService Service

The PartnerSupplierMediator mediator service obtains a quote from the PartnerSupplierService service:

To create the mediator:

1. From the Component Palette, drag a Mediator service into the middle lane of the SOA Composite Editor.

The Create Mediator window appears.

2. In the Name field, enter PartnerSupplierMediator.

3. From the Template list, select Interface Definition from WSDL.

4. Deselect the Create Composite Service with SOAP Bindings.

5. From the WSDL URL field, generate a WSDL file:
   a. Click the Generate WSDL from schema(s) icon.

   The Create WSDL dialog displays.

   b. In the URL field, click the browse for schema file icon.

   The Type Chooser dialog displays.

   c. Expand Project Schema Files > OrderInfoVOSDO.xsd and select orderInfoVOSDO.

   d. Click OK to close the Type Chooser and return to the Create WSDL dialog.

   e. In the Namespace field, enter http://www.globalcompany.example.com/ns/PartnerSupplierMediator.
f. Click the **Callback** tab.

g. In the **URL** field, click the **browse for schema file** icon.

   The Type Chooser dialog displays.

h. Expand **Project Schema Files > InternalWarehouseService.xsd** and select **WarehouseResponse**.

i. Click **OK** in the Type Chooser dialog.

j. Click **OK** in the Create WSDL dialog and return to the Create Mediator dialog.

6. In the Create Mediator dialog, deselect the **Create Composite Services with SOAP Bindings** option, and then click **OK** to create the mediator with the specified settings.

   ![Create Mediator dialog](image)

   The **PartnerSupplierMediator** mediator displays in the SOA Composite Editor.
6.2.7 Task 7: Create Routing Rules Between the PartnerSupplierMediator Mediator to the ExternalPartnerSupplier Service

To route order information from a mediator to a service, you set up routing rules.

To create a routing rule from the PartnerSupplierMediator mediator to the PartnerSupplier service:

1. Drag a wire from PartnerSupplierMediator to the PartnerSupplierService reference handle.

2. Double-click PartnerSupplierMediator to view the Mediator Editor.
For an overview of the Mediator Editor, see Oracle Fusion Middleware Developer’s Guide for Oracle SOA Suite.

3. Modify the payload transformation so that the PartnerSupplierService service receives the proper information from the mediator.

   a. Click the transformation icon next to the Transform Using field.

   The Request Transformation Map dialog displays.

   b. Select Create New Mapper File and leave the default file entry as orderItemsInfoVOSDO_To_WarehouseProcessRequest.xsl, and then click OK.

   The Data Mapper displays.
c. On the Source:PartnerSupplierMediator.wsdl (left) side, click and drag OrderTotal to orderTotal on the XSLT File:ExternalPartnerSupplier.wsdl (right) side.

d. On the Source:PartnerSupplierMediator.wsdl (left) side, expand OrderItemsInfoVO.

e. On the Source:PartnerSupplierMediator.wsdl (left) side, click and drag OrderItemsInfoVO to orderItems on the XSLT File: ExternalPartnerSupplier.wsdl (right) side.

The Auto Map Preferences dialog prompts to perform automatic mapping of the node.

f. Leave the default settings and click OK.

g. In the Data Mapper expand for-each > orderItems to see the mappings from OrderItemsInfoVO to orderItems.
h. Select **Save All** from the **File** main menu to save your work.

i. Click **X** in the **OrderInfoVOSDO_To_WarehouseProcessRequest.xsl** tab to close the Data Mapper.

j. With the **PartnerSupplier.mplan** tab back in focus, in the **Routing Rules** section, you should see file **orderInfoVO_To_WarehouseProcessRequest.xsl** in the **Transform Using** field.

4. Modify the **callback** transformation so that the **PartnerSupplierMediator** mediator receives the proper information from the **PartnerSupplierService** service:
   
a. Click the transformation icon next to the **Transform Using** field.
   
   The Request Transformation Map dialog displays.

b. Select **Create New Mapper File** and leave the default file entry as **WarehouseProcessResponse_To_WarehouseProcessResponse.xsl**, and then click **OK**.

   The Data Mapper displays.

c. On the **Source:ExternalPartnerSupplier.wsdl** (left) side, click and drag **OrderTotal** to **orderTotal** on the **XSLT File:PartnerSupplierMediator.wsdl** (right) side.
d. On the **Source:ExternalPartnerSupplier.wsdl** (left) side, click and drag `client:deliveryDate` to `deliveryDate` on the XSLT File:`PartnerSupplierMediator.wsdl` (right) side. The mappings in the Data Mapper should look like the following:

![Data Mapper screenshot]

5. Click X in the **PartnerSupplier.mplan** tab to close Mediator Editor.

6.2.8 Task 8: Wire **OrderProcessor** to the **PartnerSupplierMediator** Mediator

1. Click the **composite.xml** tab.
2. Drag a wire from **OrderProcessor** to the **PartnerSupplierMediator** reference handle.

![Mediator Editor screenshot]

3. Click the **OrderProcessor.bpel** tab to see **PartnerSupplierMediator** added to the **Partner Links** lane as a service.
4. Select **Save All** from the **File** main menu to save your work.

### 6.2.9 Task 9: Add the gWarehouseQuotes Variable

In this task, you create the `gWarehouseQuotes` variable, which provides output from the `Scope_RetrieveQuotes` scope and input to the `Scope_SelectPreferredSupplier` scope. This variable provides the response for the warehouses, including the warehouse name, delivery date, and order total.

To create this variable:

1. In the canvas workspace for the **OrderProcessor** BPEL process, click the **Variables** icon.
   
The Variables dialog displays.

2. Click the **Add** icon to add a variable.
   
The Create Variable dialog displays.

3. In the **Name** field, enter `gWarehouseQuotes`.

4. In the **Type** section, select **Element** and then select the **Browse** icon to the right of the **Element** field.
   
The Type Chooser dialog appears with a list of available services.

5. Expand **Project WSDL Files** > **InternalWarehouseService.wsdl** > **Inline Schemas** > **Schema** and select **WarehouseList**.
6. Click OK in the Type Chooser dialog.
7. Click OK in the Create Variable dialog.

The Variables dialog updates with the gWarehouseQuotes variable.

8. Click OK in the Variables dialog.

6.2.10 Task 10: Add Activities to Obtain a Quote from the InternalWarehouse Process

To add activities to obtain a quote:

1. Create an invoke activity to invoke the InternalWarehouse BPEL process.
   a. From the Component Palette, drag an Invoke activity into left-side sequence of the switch in the Scope_RetrieveQuotes scope.
b. Rename this activity by double-clicking name underneath the icon. Do not double-click the invoke icon itself.

c. In the edit field, change the name to `Invoke_InternalWarehouse`.

d. Drag the mouse from the right side of `Invoke_InternalWarehouse` to the `InternalWarehouse` BPEL process.

The Edit Invoke dialog appears and is automatically filled in with the following information:

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td><code>InvokeInternalWarehouse</code></td>
</tr>
</tbody>
</table>
| Partner Link  | `InternalWarehouseService.internalwar

houseservice_client` |
| Operation     | `WarehouseRequest`                                                   |

e. Click the Automatically Create Input Variable icon. It is the first icon to the right of the Input field.

The Create Variable dialog appears for the input variable.

f. Enter and select the following values:

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td><code>lInternalWarehouseInputVariable</code></td>
</tr>
<tr>
<td>Global Variable/Local Variable</td>
<td>Local Variable, because this variable is not needed for other scopes in the process</td>
</tr>
</tbody>
</table>

g. Click OK to close the Create Variable dialog

The Edit Invoke dialog populates with the variable in the Input field.

h. In the Edit Invoke dialog, click OK to save the variable setting.

2. Assign data to the input variable for the InternalWarehouse process:

a. Drag an Assign activity from the Component Palette section to above the `InvokeInternalWarehouse` invoke activity.

b. Rename this activity by double-clicking name underneath the icon. Do not double-click the assign icon itself.

c. In the edit field, enter `Assign_InternalWarehouseRequest`.

d. Double-click the `Assign_InternalWarehouseRequest` activity.
The Assign dialog displays.

e. From the dropdown list, select **Copy Operation**.

   The Create Copy Operation dialog appears.

   The Copy Operation tab in the Assign dialog updates to show the rule.

   

g. Click **OK** to close the Create Copy Operation dialog.

   In the Assign dialog, click **OK**.

3. Create a receive activity to receive the delivery date from the internal warehouse:

   a. From the Component Palette, drag and drop a **Receive** activity below the
      **InvokeInternalWarehouse** activity.

   b. Rename this activity by double-clicking name underneath the icon. Do not
      double-click the receive icon itself.

   c. In the edit field, change the name to **Receive_InternalWarehouse**.

   d. Drag the mouse from the right side of **ReceiveInternalWarehouse** to the
      **InternalWarehouse** BPEL process.

      The Edit Receive dialog appears and is automatically filled in with the
      following information:

      | Element    | Value                                           |
      |------------|-------------------------------------------------|
      | Name       | Receive_InternalWarehouse                       |
      | Partner Link| InternalWarehouseService.internalwarhoseservice_client |
      | Operation  | WarehouseResponse                               |

   

e. Click the **Auto-Create Request** icon. It is the first icon to the right of the
   **Variable** field.

   The Create Variable dialog appears for the input variable.

   f. Enter and select the following values:
Creating the Scope_RetrieveQuotes Flow

Creating the Second Half of the OrderProcessor BPEL Process

4. Assign data from the lInternalWarehouseResponseVariable variable to the gWarehouseQuotes variable:
   a. Drag an Assign activity from the Component Palette section to below the Receive_InternalWarehouse receive activity.
   b. Rename this activity by double-clicking name underneath the icon. Do not double-click the assign icon itself.
   c. In the edit field, enter Assign_InterWHResponse.
   d. Double-click the Assign_InterWHResponse activity.
      The Assign dialog displays.
   e. From the dropdown list, select Append Operation.
      The Create Append Operation dialog appears.
   f. Select the following values:

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>From</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Variable</td>
</tr>
<tr>
<td>Variable</td>
<td>Expand Scope - Scope_RetrieveQuotes &gt; Variables &gt; lInternalWarehouseResponseVariable &gt; payload and select ns1:WarehouseResponse. The namespace number values (for example, ns1, ns2) can vary.</td>
</tr>
<tr>
<td>To</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Variable</td>
</tr>
<tr>
<td>Variable</td>
<td>Expand Variables &gt; gWarehouseQuotes &gt; WarehouseList and select ns1:WarehouseList.</td>
</tr>
</tbody>
</table>

g. Click OK to close the Create Append Operation dialog.
   The Copy Operation tab in the Create Append Operation dialog updates to show the operation.

h. In the Assign dialog, click OK.

5. Click the Collapse (-) icon to minimize the left-side flow sequence.

6. Select Save All from the File main menu to save your work.
6.2.11 Task 11: Add Activities to Obtain a Quote from the PartnerSupplierMediator

To add activities to obtain a quote:

1. Create an invoke activity to invoke the PartnerSupplierMediator mediator:
   a. From the Component Palette, drag an Invoke activity into right-side sequence.
   b. Rename this activity by double-clicking name underneath the icon. Do not double-click the invoke icon itself.
   c. In the edit field, change the name to Invoke_PartnerSupplier.
   d. Drag the mouse from the right side of Invoke_PartnerSupplier to the PartnerSupplierMediator service.

   The Edit Invoke dialog appears and is automatically filled in with the following information:

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Invoke_PartnerSupplier</td>
</tr>
<tr>
<td>Partner Link</td>
<td>PartnerSupplierMediator.PartnerSupplierMediator</td>
</tr>
<tr>
<td>Operation</td>
<td>execute</td>
</tr>
</tbody>
</table>

   e. Click the Automatically Create Input Variable icon. It is the first icon to the right of the Input field.

   The Create Variable dialog appears for the input variable.

   f. Enter and select the following values:

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>lPartnerSupplierInputVariable</td>
</tr>
<tr>
<td>Global Variable/Local Variable</td>
<td>Local Variable, because this variable is not needed for other scopes in the process</td>
</tr>
</tbody>
</table>

   g. Click OK to close the Create Variable dialog.

   The Edit Invoke dialog populates with the variable in the Input Variable field.

   h. In the Edit Invoke dialog, click OK to save the variable setting.

2. Assign data to the input variable the PartnerSupplierService service:
   a. Drag and drop an Assign activity from the Component Palette section to above the Invoke_PartnerSupplier invoke activity.
   b. Rename this activity by double-clicking name underneath the icon. Do not double-click the assign icon itself.
   c. In the edit field, enter Assign_PartnerRequest.
   d. Double-click the Assign_PartnerRequest activity.

      The Assign dialog displays.

   e. From the dropdown list, select Copy Operation.

      The Create Copy Operation dialog appears.
f. Select the following values:

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>From</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Variable</td>
</tr>
<tr>
<td>Variable</td>
<td>Expand Variables &gt; gOrderInfoVariable and select ns4:orderInfoVOSDO. The namespace number values (for example, ns1, ns2) can vary.</td>
</tr>
<tr>
<td>To</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Variable</td>
</tr>
<tr>
<td>Variable</td>
<td>Expand Scope - RetrieveQuotes &gt; Variables &gt; lPartnerSupplierInputVariable &gt; request and select ns4:orderInfoVOSDO.</td>
</tr>
</tbody>
</table>

g. Click OK to close the Create Copy Operation dialog.

The Copy Operation tab in the Assign dialog updates to show the rule.

h. In the Assign dialog, click OK.

3. Create a receive activity to receive the delivery date from the external partner warehouse:

a. From the Component Palette, drag a receive activity below the Invoke_PartnerSupplier activity.

b. Rename this activity by double-clicking name underneath the icon. Do not double-click the invoke icon itself.

c. In the edit field, change the name to Receive_PartnerResponse.

d. Drag the mouse from the right side of Receive_PartnerResponse to the PartnerSupplierService service.

The Edit Receive dialog appears and is automatically filled in with the following information:

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Receive_PartnerResponse</td>
</tr>
<tr>
<td>Partner Link</td>
<td>PartnerSupplierMediator.PartnerSupplierMediator</td>
</tr>
<tr>
<td>Operation</td>
<td>callback</td>
</tr>
</tbody>
</table>

e. Click the Add icon. It is the first icon to the right of the Variable field.

The Create Variable dialog appears for the input variable.

f. Enter and select the following values:

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>lPartnerResponseVariable</td>
</tr>
<tr>
<td>Global Variable/Local</td>
<td>Local Variable</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Creating the Scope_SelectPreferredSupplier Scope

6.3 Creating the Scope_SelectPreferredSupplier Scope

The Scope_SelectPreferredSupplier scope uses the EvaluatePreferredSupplierRule to select the warehouse with the lowest order total.

To create the Scope_SelectPreferredSupplier scope, perform the following tasks:

g. Click OK to close the Create Variable dialog.
   The Edit Receive dialog populates with the variable in the Variable field.

h. In the Edit Receive dialog, click OK to save the variable settings.

4. Assign data from lPartnerResponseVariable variable to the gWarehouseQuotes variable:
   a. Drag an Assign activity from the Component Palette section to below the Receive_PartnerResponse receive activity.
   b. Rename this activity by double-clicking name underneath the icon. Do not double-click the assign icon itself.
   c. In the edit field, enter Assign_PartnerWHResponse.
   d. Double-click the Assign_PartnerWHResponse activity.
      The Assign dialog displays.
   e. From the dropdown list, select Append Operation.
      The Create Append Operation dialog appears.
   f. Select the following values:

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>From</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Variable</td>
</tr>
<tr>
<td>Variable</td>
<td>Expand Scope - Scope_RetrieveQuotes &gt; Variables &gt; lPartnerResponseVariable &gt; callback and select ns1:WarehouseResponse. The namespace number values (for example, ns1, ns2) can vary.</td>
</tr>
<tr>
<td>To</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Variable</td>
</tr>
<tr>
<td>Variable</td>
<td>Expand Variables &gt; gWarehouseQuotes and select ns1:WarehouseList.</td>
</tr>
</tbody>
</table>

g. Click OK to close the Create Append Operation dialog.
   The Copy Operation tab in the Create Append Operation dialog updates to show the rule.

h. In the Assign dialog, click OK.

5. Click the Collapse (-) icon to minimize the right-side flow sequence.

6. Click the Collapse (-) icon to minimize the Scope_RetrieveQuotes scope.

7. Select Save All from the File main menu to save your work.
6.3.1 Task 1: Create the Scope_SelectPreferredSupplier Scope

The Scope_SelectSupplier scope invokes the RequiresApprovalRule business rule, as shown in Figure 6–6.

Figure 6–6 Scope_SelectSupplier Scope

To add the scope:

1. From the Component Palette, drag a Scope activity below the Scope_RetrieveQuotes scope.
2. Rename this activity by double-clicking the name underneath the icon. Do not double-click the activity icon itself.
3. In the edit field, change the name to Scope_SelectPreferredSupplier.
4. Click the Expand (+) icon to expand the Scope_SelectPreferredSupplier scope.
5. Select Save All from the File main menu to save your work.

6.3.2 Task 2: Add the gPreferredSupplier Variable

In this task, you create the gPreferredSupplier variable, which provides output from the business rule variable com_example_globalcompany_ns_internalwarehouse_WarehouseResponse_o.

To create this variable:

1. In the workspace for the OrderProcessor BPEL process, click the Variables icon. The Variables dialog displays.
2. Click the Create icon to add a variable. The Create Variable dialog displays.
3. In the Name field, enter gPreferredSupplier.
4. In the **Type** section, select **Element** and then select the **Browse Elements** icon to the right of the **Element** field.

The Type Chooser dialog appears with a list of available services.

5. Expand **Project Schema Files > InternalWarehouse.xsd** and select **WarehouseResponse**. You previously imported this schema file when you created input and output variables for the **OrderProcessor** BPEL process.

6. Click **OK** to close the Type Chooser dialog.

7. In the Create Variable dialog, click **OK**.

The Variables dialog updates with the **gPreferredSupplier** variable.

8. Click **OK** in the Variables dialog.

### 6.3.3 Task 3: Create the **EvaluatePreferredSupplierRule** Business Rule

To create the business rule:

1. Click the **composite.xml** tab to view the SOA Composite Editor.

2. Drag a **Business Rule** service component into the SOA Composite Editor.

The Create Business Rule dialog displays.

3. In the **Name** field, enter **EvaluatePreferredSupplierRule** to be the name of the Oracle Business Rules dictionary.

4. Leave the default for the **Package** field.

5. In the **Inputs/Outputs** section, from the **Add** menu, select **Input** to select the input for the business rule:

The Type Chooser dialog displays.

6. In the **Inputs/Outputs** section, select the input for the business rule:

   a. From the **Add** menu, select **Input**.

      The Type Chooser dialog displays.

   b. Expand **InternalWarehouse.xsd** and select **WarehouseList**.

   c. Click **OK** to return to the Create Business Rules dialog.

7. In the **Inputs/Outputs** section, select the output for the business rule:

   a. From the **Add** menu, select **Output**.

      The Type Chooser dialog displays.

   b. Expand **Warehouse.xsd** and select **WarehouseProcessResponse**. You previously imported this schema file when you created input and output variables for the **ExternalPartnerSupplier** BPEL process.

   c. Click **OK** to return to the Create Business Rules dialog.

8. Click the **Advanced** tab and modify the name in the **Service Name** field to **EvaluatePreferredSupplierRule_DecisionService_1**. Oracle JDeveloper creates a WSDL for the business rule based on this name.

9. Click **OK** to create the business rule.

The **EvaluatePreferredSupplierRule** business rule displays in the composite.
10. Select Save All from the File main menu to save your work.

6.3.4 Task 4: Reference the RequiresApprovalRule Dictionary in the BPEL Designer

To reference the EvaluatePreferredSupplierRule in the Scope_SelectSupplier scope:

1. Click the OrderProcessor.bpel tab.
2. From the Component Palette, drag a Business Rule activity into the Scope_SelectSupplier scope.
   The Business Rule dialog displays.
3. In the Name field, enter BusinessRule_SelectPreferredSupplier.
4. From the Dictionary list, select EvaluatePreferredSupplierRule, which is the rule you created in Section 6.3.3, "Task 3: Create the EvaluatePreferredSupplierRule Business Rule."
5. Leave the default settings for the Service and Operation fields.
6. Create input to the business rule, so that the gWarehouseQuotes variable assigns data to input variable com_example_globalcompany_ns_internalwarehouse_WarehouseList_i for the business rule.
   a. In the Assign Input Facts tab, click the Create icon.
      The Decision Fact Map dialog displays.
   b. Select the following values:
c. Click OK.

The copy operation displays in the Assign Input Facts tab of the Business Rule dialog.

7. Create output from the business rule, so that the variable `com_example_globalcompany_ns_internalwarehouse_WarehouseResponse_o` sends data to the `gPreferredSupplier` variable:

   a. Click the Assign Output Facts tab.

   b. In the Assign Output Facts tab, click the Create icon.

   The Decision Fact Map dialog displays.

   c. Select the following values:

```
<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>From</td>
<td></td>
</tr>
<tr>
<td></td>
<td>■ Type Variable</td>
</tr>
<tr>
<td></td>
<td>■ Variable Expand Variables &gt; gWarehouseQuotes and select ns3:WarehouseList.</td>
</tr>
<tr>
<td></td>
<td>Note: The namespace number values (for example, ns1, ns2) can vary.</td>
</tr>
<tr>
<td>To</td>
<td></td>
</tr>
<tr>
<td></td>
<td>■ Type Business Rule Facts</td>
</tr>
<tr>
<td></td>
<td>■ Variable Expand com_example_globalcompany_ns_internalwarehouse_WarehouseList_i and select ns10:WarehouseList.</td>
</tr>
</tbody>
</table>
```

c. Click OK.

The copy operation displays in the Assign Input Facts tab of the Business Rule dialog.


9. Select Save All from the File main menu to save your work.
6.3.5 Task 5: Add a New Rule for Ruleset in Rules Designer

In this task, you add the Rule_FindPreferredSupplier rule. This rule uses the gWarehouseQuotes variable as input, which provides the order total for the warehouses. The logic for the rule selects the warehouse as the current supplier with the lowest order total, as shown in Figure 6–7.

**Figure 6–7  Rule_FindPreferredSupplier Rule**

To create the rule:

1. In Rules Designer, select Ruleset_1 from the left menu.
2. From the Create dropdown list, select Create Rule.
   A new rule named Rule_1 displays.
3. Select Rule_1 and modify the name to Rule_FindPreferredSupplier.
4. Click <insert test> to display the statement template.
5. For the IF condition, click the left-hand operand and expand WarehouseList > warehouseResponse and select size().
Creating the Scope SelectPreferredSupplier Scope

6. Click the operator and select >.

b. Click the right-hand operand and enter 0.

6. Click the Show Advanced Settings icon:

7. Click the Advanced Mode option, which enables you to select advanced actions for the THEN true clause.

8. For the THEN true clause, perform the following steps to complete the first statement:
   a. Click <insert action> and select assign new.
   b. Click <type> and select int.
   c. Click var and enter index.
   d. Click <expression> and enter 0.

9. Create the second statement as:
   assign new WarehouseResponse preferredSupplier = new WarehouseResponse()

10. Perform the following steps to complete the third statement:
   a. Click <insert action> and select while.
   b. Click <boolean expression> and enter:
      index < WarehouseList.warehouseResponse.size()
   c. Under the while statement, create the following assign new activity:
      assign new WarehouseResponse currentSupplier=(WarehouseResponse)WarehouseList.warehouseResponse.get(index)
   d. Under the while statement, click <insert action> and select if.
   e. Click <boolean expression> and click the Expression Builder icon.
   f. In the Expression Builder, enter the following:
      currentSupplier.orderTotal < preferredSupplier.orderTotal ||
preferredSupplier.orderTotal <= 0

You can select currentSupplier.orderTotal and preferredSupplier.orderTotal from the Options and click Insert into Expression. Click OK in the Expression Builder.

g. Under the if statement, click <insert action> and select modify.

h. Click <target> and select preferredSupplier.

i. Click <add property>. The Properties dialog displays.

j. From the Value dropdown menu for deliveryDate, orderTotal, and warehouse, select currentSupplier.deliveryDate, currentSupplier.orderTotal, and currentSupplier.warehouse, respectively.

k. Click Close in the Properties dialog.

l. Under the while statement, create the following assign statement:
   assign index = index + 1

11. Create the fourth statement as:
    assert preferredSupplier

12. Select Save All from the File main menu to save your work.

13. Click X in the EvaluatePreferredSupplierRule.rules tab to close the Rules Designer.

### 6.4 Creating the Services and Routing Required for the Scope_FulfillOrder Scope

After a warehouse is selected for an order, the OrderProcessor BPEL process invokes the Scope_FulfillOrder scope, which uses the FulfillOrder mediator service to determine where orders are forwarded for fulfillment. It sends the order to a file adapter named USPSShipment and a JMS adapter named FulfillmentBatch. For this tutorial, the order is just sent to a JMS queue. There is no consumer of the order data from the queue. This tutorial has you create the JMS adapter, so you can learn how to send messages to a JMS adapter.

Figure 6–8 shows the flow of the FulfillOrder mediator service in the SOA Composite Editor.
Prior to creating the `Scope_FulfillOrder` scope in the `OrderProcessor` BPEL process, you create the following:

- FulfillOrder mediator service
- USPSShipment file adapter
- FulfillmentBatch JMS adapter

The tasks to create these elements are as follows:

- Section 6.4.1, "Task 1: Create USPSShipment File Adapter"
- Section 6.4.2, "Task 2: Create FulfillmentBatch JMS Service"
- Section 6.4.3, "Task 3: Create FulfillOrder Mediator Service Component"
- Section 6.4.4, "Task 4: Create Routing Rules"
- Section 6.4.5, "Task 5: Wire OrderProcessor to FulfillOrder Mediator Service"

### 6.4.1 Task 1: Create USPSShipment File Adapter

The USPSShipment adapter writes order information to a flat file in the C:\tmp directory.

To create this adapter:

1. Copy `LegacyOrderBookingPO.xsd` from `DEMO_DOWNLOAD_HOME\CompositeServices\OrderBookingComposite\xsd` to directory `MY_FOD_HOME\OrderBookingComposite\xsd`, so you can reference it when you create the adapter.
2. In the Application Navigator, click the Refresh icon.
3. From the Component Palette, drag and drop the File Adapter into the right swim lane of the SOA Composite Editor.
   The Adapter Configuration Wizard displays.
4. Click Next.
   The Service Name page displays.
5. In the Service Name field, enter USPSShipment, and then click Next.
   The Operation page displays.
6. In the Adapter Interface page, select Define from operation and schema, and then click Next.
   The Operation page displays.

7. In Operation Type section, click Write File, and then click Next.
   When you select Write File, the Operation Name is automatically set to Write.
   The File Configuration page displays.

8. Enter and select the following values:

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Directory for Outgoing Files</td>
<td>Click Browse to find the directory where you want the adapter to write the files. For example, you can select /tmp.</td>
</tr>
<tr>
<td>File Naming Convention</td>
<td>Specify how you want to name the files. Enter po_%SEQ%.txt. The %SEQ% indicates that the file names are to be numbered sequentially.</td>
</tr>
<tr>
<td>Number of Messages Equals</td>
<td>You can leave it at 1, which means that each order writes to a separate file.</td>
</tr>
</tbody>
</table>

9. Accept the other default settings, and then click Next.
   The Messages page displays.

10. Click the Browse icon to the right of the URL field.
    The Type Chooser dialog displays.

11. Expand Project Schema Files > LegacyOrderBookingPO.xsd and select PurchaseOrder.

12. Click OK to return to the Message page.
    The URL field fills in with value xsd/LegacyOrderBooking.xsd and the Schema Element field fills in with value PurchaseOrder.

13. Click Next.
    The Finish page displays.

14. Click Finish.
    The SOA Composite Editor updates with USPSShipment adapter. In addition, the OrderBookingComposite directory updates with files USPSShipment_file.jca and USPSShipment.wsdl for the adapter, which you can also see in the Application Navigator.

15. Select Save All from the File main menu to save your work.

6.4.2 Task 2: Create FulfillmentBatch JMS Service

The FulfillmentBatch JMS adapter sends the order information to a JMS server.

To create this adapter:

1. From the Component Palette, drag a JMS Adapter from the Service Adapters list into the right swim lane of the SOA Composite Editor.
   The Adapter Configuration Wizard displays.

2. In the Welcome page, click Next.
Creating the Services and Routing Required for the Scope_FulfillOrder Scope

3. In the **Service Name** field, enter FulfillmentBatch, and then click **Next**.

   The JMS Provider page displays.

4. From the **Oracle Enterprise Messaging Service (OEMS)** list, select **Oracle WebLogic JMS**, and then click **Next**.

   The Service Connection page displays.

5. From the **Connection** list, select the **MyAppServerConnection** running Oracle SOA Suite, and then click **Next**.

   The Adapter Interface page displays.

6. Select **Define from operation and schema**, and then click **Next**.

   The Operation page displays.

7. Select **Produce Message**, and then click **Next**.

   The Produce Operation Parameters page displays.

8. Click the **Browse** icon next to the **Destination Name** field.

   The Select Destination dialog displays.

9. In the **Destination** section, expand **SOAJMSModule** and select **DemoSupplierTopic**, and then click **OK**.

   If you ran the **seedFodJmsResource** script in Section 2.1.3, "Task 3: Deploy the WebLogic Fusion Order Demo Application," to run the sample application, this topic exists. If topic **DemoSupplierTopic** is not available, create it through the Oracle WebLogic Server Administration Console, following these procedures:

   - **Section A.1, "Task 1: Create the JMS Topic"
   - **Section A.2, "Task 2: Create the JMS Topic Connection Factory"
   - **Section A.3, "Task 3: Add the Connection Pool"

10. In the **JNDI Name** field, enter **eis/Jms/TopicConnectionFactory**.

    The Messages page displays.

11. Accept the other defaults in the page, and click **Next**.

12. In the Messages page, click option **Native format translation is not required (Schema is Opaque)**, and then click **Next**.

    The Finish page displays.

13. Click **Finish**.

    The SOA Composite Editor updates with **FulfillmentBatch** adapter. In addition, the **OrderBookingComposite** directory updates with files **FulfillmentBatch_jms.jca** and **FulfillmentBatch.wsdl** for the adapter, which you can also see in the Application Navigator.

14. Select **Save All** from the **File** main menu to save your work.

---

### 6.4.3 Task 3: Create FulfillOrder Mediator Service Component

To create the **FulfillOrder** mediator service component:
1. From the Component Palette, drag a Mediator service into the designer of the SOA Composite Editor.

   The Create Mediator window appears.

2. In the Name field, enter FulfillOrder.

3. From Template, select Interface Definition from WSDL.

4. Deselect the Create Composite Service with SOAP Bindings.

5. From the WSDL URL field, generate a WSDL file:
   a. Click the Generate WSDL from schema(s) icon.
      The Create WSDL dialog displays.
   b. In the URL field, click the browse for schema file icon.
      The Type Chooser dialog displays.
   c. Expand Project Schema Files > OrderInfoVOSDO.xsd and select orderInfoVOSDO.
   d. Click OK.
   e. In the Create WSDL dialog, accept the default values for the remaining fields, and click OK.

6. In the Create Mediator dialog, click OK to create the mediator with the settings.

   The FulfillOrder mediator displays in the SOA Composite Editor.
7. Select Save All from the File main menu to save your work.

6.4.4 Task 4: Create Routing Rules

For the WebLogic Fusion Order Demo application, you create routing rules from the FulfillOrder mediator to the USPSShipmen and FulfillmentBatch adapters:

To create a routing rule from the FulfillOrder mediator service to the FullmentBatch JMS adapter:

1. Drag a wire from the FulfillOrder mediator to the USPSShipmen reference handle.

2. Drag a wire from the FulfillOrder mediator to the FulfillmentBatch reference handle.

3. Double-click FulfillOrder to view the Mediator Editor.
4. Modify the transformation for FulfillmentBatch JMS service:
   a. Click the transformation icon next to the Transform Using field.
      The Request Transformation Map dialog displays.
   b. Select Create New Mapper File and leave the default file entry as orderInfoVOSD_To_opaqueElement.xsl as the file name, and then click OK.
      The Data Mapper displays.
   c. On the Source:FulfillOrderRef.wsdl (left) side, click and drag inp1:orderInfoVOSDO to opaque:opaqueElement on the XSLT File:FulfillmentBatch.wsdl (right) side.
   d. From the Component Palette, expand the String Functions list.
   e. Drag the get-content-as-string function on the top of the mapping and connect the source and target to it.
      This function returns the XML representation of the orderInfoVOSD input. The Data Mapper dialog should look like the following now.
   f. Click X in the orderInfoVODO_To_opaqueElement.xsl tab to close the Data Mapper.

5. Modify the transformation for the USPSShipment adapter:
   a. Click the transformation icon next to the Transform Using field.
      The Request Transformation Map dialog displays.
b. Select **Create New Mapper File** and leave the default file entry as `orderInfoVOSDO_To_PurchaseOrder.xsl` as the file name, and then click **OK**.

The Data Mapper displays.

c. On the Source:`FulfillOrder.wsdl` (left) side, click and drag the following from `orderInfoVOSDO` to `PurchaseOrder` on the XSLT File:`USPSShipment.wsdl` (right) side:
- **OrderId** to **ID**
- **OrderDate** to **OrderInfo > OrderDate**
- **OrderStatusCode** to **OrderInfo > OrderStatus**
- **OrderTotal** to **OrderInfo > OrderPrice**
- **CustomerId** to **CustID**
- **ShipToName** to **ShipTo > Name > Last**
- **ShipToPhoneNumber** to **UserContact > PhoneNumber**
- **Address1** to **ShipTo > Street**
- **City** to **ShipTo > City**
- **Postalcode** to **ShipTo > Zip**
- **StateProvince** to **ShipTo > State**
- **CountryId** to **ShipTo > Country**
- **OrderItemsInfoVO** to **Item**

For the **OrderItemsInfoVO** to **OrderItems**, the Auto Map Preferences dialog displays. Leave the default settings and click **OK**.

d. Click **X** in the **orderInfoVOSDO_To_PurchaseOrder.xsl** tab to close the Data Mapper.

e. Click **X** in the **FulfillOrder.mplan**.

f. Select **Save All** from the **File** main menu to save your work.

### 6.4.5 Task 5: Wire OrderProcessor to FulfillOrder Mediator Service

To wire the OrderProcessor BPEL process to the FulfillOrder mediator:

1. Drag a wire from **OrderProcessor** to the **FulfillOrder** reference handle.
2. Select **Save All** from the **File** main menu to save your work.

3. Click the **OrderProcessor.bpel** tab to see the **FulfillOrder** mediator service added to the main process.

### 6.5 Creating the Scope_FulfillOrder Scope

To create the **Scope_FulfillOrder** scope in the **OrderProcessor** process, perform the following tasks:

- **Task 1: Add the Scope_FulfillOrder Scope**
- **Task 2: Create the InvokeFulfillOrder Activity**
- **Task 3: Create the AssignFulfillRequest Activity**

#### 6.5.1 Task 1: Add the Scope_FulfillOrder Scope

To create the **Scope_FulfillOrder** scope:

1. From the Component Palette, drag a **Scope** activity below the **Scope_SelectPreferredSupplier** activity.
2. Rename this activity by double-clicking the name underneath the icon. Do not double-click the activity icon itself.
3. In the edit field, change the name to **Scope_FulfillOrder**.
4. Click the **Expand (+)** icon to expand the **Scope_FulfillOrder** scope.

#### 6.5.2 Task 2: Create the InvokeFulfillOrder Activity

To create an invoke activity to call the **FulfillOrder** mediator service:

1. From the Component Palette, drag and drop an **Invoke** activity into the **Scope_FulfillOrder** scope.
2. Rename this activity by double-clicking name underneath the icon. Do not double-click the invoke icon itself.

3. In the edit field, change the name to Invoke_FulfillOrder.

4. Drag the mouse from the right side of Invoke_FulfillOrder activity to the FulfillOrder service.

The Edit Invoke dialog appears and is automatically filled in with the following information:

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Invoke_Fulfillorder</td>
</tr>
<tr>
<td>Partner Link</td>
<td>FulfillOrder.FulfillOrder</td>
</tr>
<tr>
<td>Operation</td>
<td>execute</td>
</tr>
</tbody>
</table>

5. Click the Automatically Create Input Variable icon. It is the first icon to the right of the Input Variable field.

The Create Variable dialog appears for the input variable.

6. Enter and select the following values:

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Modify the name of the input variable to lFulfillOrder_InputVariable.</td>
</tr>
<tr>
<td>Global Variable/Local Variable</td>
<td>Select Local Variable.</td>
</tr>
</tbody>
</table>

7. Click OK to close the Create Variable dialog.

The Edit Invoke dialog populates with the variable in the Input Variable field.

8. In the Edit Invoke dialog, click OK to save the variable setting.
6.5.3 Task 3: Create the AssignFulfillRequest Activity

To assign data to the input variable for FulfillOrder mediator service:

1. Drag and drop an Assign activity from the Component Palette section to above the Invoke_FulfillOrder invoke activity.
2. Rename this activity by double-clicking name underneath the icon. Do not double-click the assign icon itself.
3. In the edit field, enter Assign_OrderData.
4. Double-click the Assign_OrderData activity.
   The Assign dialog displays.
5. From the dropdown menu, select Copy Operation.
   The Create Copy Operation dialog appears.
6. Select the following values:

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>From</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Variable</td>
</tr>
<tr>
<td>Variable</td>
<td>Expand Variables &gt; gOrderInfoVariable and select ns4:orderInfoVOSDO. The namespace number values (for example, ns1, ns2) can vary.</td>
</tr>
<tr>
<td>To</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Variable</td>
</tr>
<tr>
<td>Variable</td>
<td>Expand Scope - Scope_FulfillOrder &gt; Variables &gt; lFulfillOrder_InputVariable &gt; request and select ns4:orderInfoVOSDO.</td>
</tr>
</tbody>
</table>

7. Click OK to close the Create Copy Operation dialog.
   The Copy Operation tab in the Assign dialog updates to show the rule.
8. In the Assign dialog, click OK.
9. Click the Collapse (-) icon to minimize the Scope_FulfillOrder scope.
10. Select Save All from the File main menu to save your work.

6.6 Creating the Scope_UpdateStatusToComplete Scope for Completed Orders

The Scope_UpdateStatusToComplete scope assigns a status of complete to the order entity variable, which updates the order in the database. If you did not use an entity variable, you would have to create a database adapter.

To create the Scope_UpdateStatusToComplete scope:

1. Add the Scope_UpdateStatusToComplete scope:
   a. From the Component Palette, drag a Scope activity below the Scope_FulfillOrder activity.
   b. Rename this activity by double-clicking the name underneath the icon. Do not double-click the activity icon itself.
c. In the edit field, change the name to Scope_UpdateStatusToComplete.

d. Click the Expand (+) icon to expand the Scope_UpdateStatusToComplete scope.

2. Create an assign activity to complete the order:
   a. From the Component Palette, drag an Assign activity into the Scope_UpdateStatusToComplete scope.
   b. Rename this activity by double-clicking name underneath the icon. Do not double-click the assign icon itself.
   c. In the edit field, enter UpdateOrderStatus.
   d. Double-click the UpdateOrderStatus activity.

   The Assign dialog displays.

3. Assign a status of complete for the order:
   a. From the dropdown list, select Copy Operation.

   The Create Copy Operation dialog appears.

   b. Select the following values:

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>From</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Expression</td>
</tr>
</tbody>
</table>
   | Expression | 1. Select the XPath Expression Builder icon to display the Expression Builder dialog.  
                                2. In the Expression box, enter the following:  
                                   string('complete')  
                                3. Click OK to return to the Create Copy Operation dialog. |
   | To      |       |
   | Type    | Variable |
   | Variable | Expand Variables > gOrderInfoVariable > ns4:orderInfoVOSDO and select ns4:OrderStatusCode.  
                                Note: The namespace number values (for example, ns1, ns2) can vary. |

   c. Click OK to close the Create Copy Operation dialog and return to the Assign dialog.

   The Copy Operation tab in the Assign dialog updates to show the rule.

4. Assign the order ID to the output variable for the OrderProcessor BPEL process:
   a. From the dropdown list, select Copy Operation.

   The Create Copy Operation dialog appears.

   b. Select the following values:
Creating the Scope_NOTIFYCustomerofCompletion Scope

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>From</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Variable</td>
</tr>
<tr>
<td>Variable</td>
<td>Expand Variables &gt; inputVariable &gt; payload &gt; ns4:WarehouseRequest and select ns4:orderId. Note: The namespace number values (for example, ns1, ns2) can vary.</td>
</tr>
<tr>
<td>To</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Variable</td>
</tr>
<tr>
<td>Variable</td>
<td>Expand Variables &gt; outputVariable &gt; payload &gt; ns3:WarehouseResponse and select Total.</td>
</tr>
</tbody>
</table>

c. Click OK to close the Create Copy Operation dialog and return to the Assign dialog.
The Copy Operation tab in the Assign dialog updates to show the rule.

5. In the Assign dialog, click OK.
6. Click the Collapse (-) icon to minimize the Scope_UpdateStatusToComplete scope.
7. Select Save All from the File main menu to save your work.

6.7 Creating the Scope_NOTIFYCustomerofCompletion Scope

The Scope_NOTIFYCustomerofCompletion scope uses the Oracle User Messaging Service to send an email to the customer when the order is fulfilled.

To create the Scope_NOTIFYCustomerofCompletion scope:

1. From the Component Palette, drag a Scope activity below the Scope_UpdateStatusToComplete activity.
2. Rename this activity by double-clicking the name underneath the icon. Do not double-click the activity icon itself.
3. In the edit field, change the name to Scope_NOTIFYCustomerofCompletion.
4. Click the Expand (+) icon to expand the Scope_NOTIFYCustomerofCompletion scope.
5. From the Component Palette, drag an Email activity into the empty scope.
The Email dialog displays.
6. In the From Account field, leave the default value as Default.
7. For the To field, specify the customer’s email address from the ConfirmedEmail parameter from the gCustomerInfoVariable variable:
   a. Click the XPath Expression Builder icon next to the To field to display the Expression Builder dialog
   b. In the BPEL Variables box, expand gCustomerInfoVariable > parameters > findCustomerInfoVO1CustomerInfoVOCriteriaResponse > result and select ConfirmedEmail.
The Content Preview box shows:
Creating the Scope_NotifyCustomerofCompletion Scope

bpws:getVariableData('gCustomerInfoVariable','parameters','/ns3:findCustomerInfoVO1CustomerInfoVOCriteriaResponse/ns3:result/ns2:ConfirmedEmail')

c. Click **Insert Into Expression**.
The Expression box updates.

d. Click **OK** to close the Expression Builder.

8. For the **Subject** field, specify *Order with id OrderID shipped*. To perform the steps:

a. In the **Subject** field box, enter *Order with id*, followed by a space.

b. Click the **XPath Expression Builder** icon next to the **Subject** field to display the Expression Builder dialog.

c. In the Expression Builder dialog, in the **BPEL Variables** box, expand *gOrderInfoVariable* > *orderInfoVOSDO* and select the *OrderID* node.

The Content Preview box shows:

bpws:getVariableData('gOrderInfoVariable','/ns2:orderInfoVOSDO/ns2:OrderId')

d. Click **Insert Into Expression**.
The Expression box updates with the *OrderId* variable.

e. Click **OK** to close the Expression Builder.

f. Append the end of the **Subject** field with *shipped*!. The entire string

Order with id

<%bpws:getVariableData('gOrderInfoVariable','/ns2:orderInfoVOSDO/ns2:OrderId')%> shipped!

9. For the **Body** field, specify *Dear FirstName, your order has been shipped*. To perform the steps:

a. In the **Subject** field box, enter *Dear*, followed by a space.

b. Click the **XPath Expression Builder** icon next to the **Subject** field to display the Expression Builder dialog.

c. In the Expression Builder dialog, in the **BPEL Variables** box, expand *gCustomerInfoVariable* > *parameters* > *findCustomerInfoVO1CustomerInfoVOCriteriaResponse* > *result* and select *FirstName*.

The Content Preview box shows:

bpws:getVariableData('gCustomerInfoVariable','parameters','/ns3:findCustomerInfoVO1CustomerInfoVOCriteriaResponse')

d. Click **Insert Into Expression**.
The Expression box updates with the *OrderId* variable.

e. Click **OK** to close the Expression Builder.

f. Append the end of the **Body** field with the following string:
Adding a Catch Branch for Incomplete Orders for the Entire Process

Add a catch branch to the process as a whole so that you can update the order status to be canceled in case an error occurs anywhere in the process. Figure 6–9 shows the catches for the OrderProcessor process. The branch on the left is a catch branch, which assigns a status of ‘FAULTED’ to an order when there is an error. The branch on the right is a catchAll branch, which catches any fault not handled by the catch branch.

Figure 6–9  Catches in OrderProcessor

To create the catches for the process, perform the following tasks:

1. Click the Add Catch Branch icon for the process, as shown in the following figure:

2. Double-click the catch to display the Catch dialog.

3. Enter the following values:

The entire string is as follows:

Dear <\%bpws:getVariableData('gCustomerInfoVariable','parameters','/ns6:findCustomerInfoVO1CustomerInfoVOCriteriaResponse/ns6:result/ns4:FirstName')\%>, your order has been shipped.

10. Click OK in the Email dialog.

11. Click the Collapse (-) icon to minimize the Scope_NotifyCustomerofCompletion scope.

12. Select Save All from the File main menu to save your work.

6.8 Adding a Catch Branch for Incomplete Orders for the Entire Process

Click OK in the Email dialog.

Click the Collapse (-) icon to minimize the Scope_NotifyCustomerofCompletion scope.

Select Save All from the File main menu to save your work.
Adding a Catch Branch for Incomplete Orders for the Entire Process

4. Click OK in the Catch dialog.

5. Click the Expand (+) icon to expand the client:OrderProcessorFault catch.

6. Create an invoke activity to invoke the orderprocessor_client service:
   a. From the Component Palette, drag an Invoke activity into right-side sequence.
   b. Rename this activity by double-clicking name underneath the icon. Do not double-click the invoke icon itself.
   c. In the edit field, change the name to Invoke_FaultCallbackOPFault.
   d. Drag the mouse from the left side of Invoke_FaultCallbackOPFault to orderprocessor_client.

   The Edit Invoke dialog appears and is automatically filled in with the following information:

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Invoke_FaultCallbackOPFault</td>
</tr>
<tr>
<td>Partner Link</td>
<td>orderprocessor_client_ep</td>
</tr>
<tr>
<td>Operation</td>
<td>processFault</td>
</tr>
</tbody>
</table>

   e. Click the Browse Variables icon. It is the first icon to the right of the Input field.

   The Variable Chooser dialog appears.

   f. Select the gOrderProcessorFaultVariable and then click OK.

   You created this variable in Section 5.10, "Creating Catch Branches for the Scope_AuthorizeCreditCard."

   g. In the Edit Invoke dialog, click OK to save the variable setting.

7. In the client:OrderProcessorFault catch, create an assign activity to assign expression 'FAULTED' as input to global variable gOrderInfoVariable.
   a. From the Component Palette, drag an Assign activity into the branch, above the Invoke_FaultCallbackOPFault.
   b. Rename this activity by double-clicking name underneath the icon. Do not double-click the assign icon itself.
   c. In the edit field, enter Assign_FaultedStatusToOrder.
   d. Double-click Assign_FaultedStatusToOrder.

   The Assign dialog displays.

   e. From the dropdown list, select Copy Operation.

   The Create Copy Operation dialog appears.

   f. Enter and select the following values:
g. Click OK to close the Create Copy Operation dialog and return to the Assign dialog. The Copy Operation tab in the Assign dialog updates to show the rule.

h. In the Assign dialog, click OK.

8. Click the Collapse (-) icon to minimize the client:OrderProcessorFault catch.

9. Click the Add CatchAll Branch icon for the process to catch any faults that are not handled by client:OrderProcessorFault catch.

10. Expand the CatchAll catch.

11. Create an invoke activity to invoke the orderprocessor_client service:
   a. From the Component Palette, drag an Invoke activity into right-side sequence.
   b. Rename this activity by double-clicking name underneath the icon. Do not double-click the invoke icon itself.
   c. In the edit field, change the name to Invoke_FaultCallback.
   d. Drag the mouse from the left side of Invoke_FaultCallback to orderprocessor_client.

The Edit Invoke dialog appears and is automatically filled in with the following information:

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Invoke_FaultCallback</td>
</tr>
<tr>
<td>Partner Link</td>
<td>orderprocessor_client</td>
</tr>
</tbody>
</table>
Adding a Catch Branch for Incomplete Orders for the Entire Process

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operation</td>
<td>processFault</td>
</tr>
</tbody>
</table>

e. Click the Browse Variables icon. It is the first icon to the right of the Input field.

   The Variable Chooser dialog appears.

f. Select the $gOrderProcessorFaultVariable$ and then click OK.

   You created this variable in Section 5.10, "Creating Catch Branches for the Scope_AuthorizeCreditCard."

g. In the Edit Invoke dialog, click OK to save the variable setting.

12. Assign expression $ora:getFaultAsString()$ to global variable $gOrderProcessorFaultVariable$.

   a. From the Component Palette, drag an Assign activity into the branch, above the Invoke_FaultCallback.

   b. Rename this activity by double-clicking name underneath the icon. Do not double-click the assign icon itself.

   c. In the edit field, enter Assign_Fault.

   d. Double-click Assign_Fault.

      The Assign dialog displays.

   e. From the dropdown list, select Copy Operation.

      The Create Copy Operation dialog appears.

   f. Select the following values:

      | Element | Value |
      |---------|-------|
      | From    |       |
      | Type    | Expression |
      | Expression | 1. Select the XPath Expression Builder icon. |
      |         | 2. In the Functions section in the lower right, from the menu, select Advanced Functions and then select getFaultAsString from the menu options. |
      |         | 3. Click Insert Into Expression, and then click OK to return to the Create Copy Operation dialog. |

      | To      |       |
      | Type    | Variable |
      | Variable | Expand Variables > $gOrderProcessorFaultVariable$ and select summary. |

   g. Click OK to close the Create Copy Operation dialog and return to the Assign dialog.

      The Copy Operation tab in the Assign dialog updates to show the rule.

13. Assign expression $ora:getFaultName()$ to global variable $gOrderProcessorFaultVariable$. 
a. From the dropdown list, select **Copy Operation**.
The Create Copy Operation dialog appears.

b. Select the following values:

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>From</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Expression</td>
</tr>
</tbody>
</table>
| Expression | 1. Select the XPath Expression Builder icon.  
            2. In the Functions section in the lower right, from the menu, select Advanced Functions and then select `getFaultName` from the menu options.  
            3. Click Insert Into Expression, and then click OK to return to the Create Copy Operation dialog. |
| To      |       |
| Type    | Variable |
| Variable | Expand Variables > `gOrderProcessorFaultVariable` and select `code`. |

c. Click **OK** to close the Create Copy Operation dialog and return to the Assign dialog.
The Copy Operation tab in the Assign dialog updates to show the rule.

d. In the Assign dialog, click **OK**.

14. Click the **Collapse** (-) icon to minimize the catch.

15. Select **Save All** from the **File** main menu to save your work.
Adding the OrderPendingEvent Mediator Service Component

Business events consist of message data sent as the result of an occurrence in a business environment. When a business event is published, other service components can subscribe to it.

In this chapter, you learn how to subscribe to a business event using Oracle Mediator. At a high-level, you perform the following tasks:

- Create a business named NewOrderSubmitted.
- Create OrderPendingEvent mediator service component to subscribe to the NewOrderSubmitted business event and initiate the OrderProcessor BPEL process through a routing rule to process the order through a routing rule.

This chapter contains the following sections:

- Section 7.1, "Task 1: Create the NewOrderSubmitted Business Event"
- Section 7.2, "Task 2: Create Mediator Service Component to Subscribe to NewOrderSubmitted Business Event"
- Section 7.3, "Task 3: Route OrderPendingEvent Mediator Service Component to OrderProcessor BPEL Process"

7.1 Task 1: Create the NewOrderSubmitted Business Event

To create the NewOrderSubmitted business event:

1. Click the composite.xml tab to view the SOA Composite Editor.
2. Launch the Event Definition Creation wizard in either of two ways:
   a. In the SOA Composite Editor, click the Event Definition Creation icon above the designer:

   ![Event Definition Creation icon](image)

   b. From the File main menu, select New > SOA Tier > Service Components > Event Definition.

   The Event Definition Creation dialog appears.
3. In the Event Definition Name field, enter OrderEO. Oracle JDeveloper saves the NewOrderSubmitted event to the orderEO.edl file.
4. Leave the default settings for the **Namespace** field.
5. Click the **Add an Event** icon to add an event.
   The Add an Event dialog appears.
6. Enter the following values:

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Element</strong></td>
<td>1. Click the <strong>Browse</strong> icon to select the payload.</td>
</tr>
<tr>
<td></td>
<td>The Type Chooser dialog displays.</td>
</tr>
<tr>
<td></td>
<td>2. Expand <strong>Project Schema Files &gt; OrderEO.xsd</strong> and select <strong>NewOrderSubmittedInfo</strong>. You imported this schema file when you copied the services in Section 5.2, as you were creating the <strong>OrderProcessor</strong> BPEL process.</td>
</tr>
<tr>
<td></td>
<td>3. Click <strong>OK</strong>.</td>
</tr>
</tbody>
</table>

| **Name**     | **NewOrderSubmitted**                                                  |

7. Click **OK**.
8. In the Event Definition Creation dialog, click **OK**.
   The Business Events Editor displays with the **NewOrderSubmitted** event.
9. Select **Save All** from the **File** main menu to save your work.
10. Click **X** in the **OrderEO.edl** tab to close the definition file.
    The business event is published to MDS and you are returned to the SOA Composite Editor.

## 7.2 Task 2: Create Mediator Service Component to Subscribe to NewOrderSubmitted Business Event

To subscribe to the **NewOrderSubmitted** business event and initiate the **OrderProcessor** BPEL process:

1. Drag a **Mediator** service component into the SOA Composite Editor. This service component enables you to subscribe to the business event.
2. In the **Name** field, enter **OrderPendingEvent**.
3. From the **Templates** list, select **Subscribe to Events**.
   The window refreshes to display an events table.
4. Click the **Subscribe to a new event** icon to display the Event Chooser dialog.
5. With the **NewOrderSubmitted** event selected, click **OK**.
   You are returned to the Create Mediator dialog.
**one and only one** specifies that events are delivered to the subscriber in its own global (that is, JTA) transaction. Any changes made by the subscriber within the context of that transaction are committed when the event processing is complete. If the subscriber fails, the transaction is rolled back. Failed events are retried a configured number of times before being delivered to the hospital queue.

$\textit{publisher}$ specifies the event requires a security role.

6. In the Create Mediator dialog, click **OK**.

The `OrderPendingEvent` mediator displays in the SOA Composite Editor. The icon on the left side that indicates that mediator is configured for an event subscription.

7. Click **Source**.

The following source code provides details about the subscribed event of the mediator service component.

```xml
<component name="OrderPendingEvent">
  <implementation.mediator src="OrderPendingEvent.mplan"/>
  <business-events>
    <subscribe xmlns:sub1="http://schemas.oracle.com/events/edl/OrderEO"
               name="sub1:NewOrderSubmitted" consistency="oneAndOnlyOne"
               runAsRoles="$publisher"/>
  </business-events>
</component>
```

### 7.3 Task 3: Route `OrderPendingEvent` Mediator Service Component to `OrderProcessor` BPEL Process

To create a routing rule from the `OrderPendingEvent` mediator service component to the `OrderProcessor` BPEL process:

1. Back in the Design tab, drag a wire from `OrderPendingEvent` to the `OrderProcessor` reference handle.
2. Double-click **OrderPendingEvent** tab to see the rule in the Mediator Editor:

3. Modify the transformation used for the **OrderPendingEvent** mediator service component so that the **OrderProcessor** BPEL process receives input from the **NewOrderSubmitted** business event:
   a. Click the transformation icon next to the **Transform Using** field.
      The Event Transformation Map dialog displays.
   b. Select **Create New Mapper File**, use the default name `NewOrderSubmitted_To_WarehouseRequest.xsl` in the accompanying field, and then click **OK**.
      The Data Mapper displays.
c. On the Source:OrderEO.xsd (left) side, click and drag OrderID to ns1:WarehouseRequest > ns1:orderId on the XSLT File: OrderProcessor.wsdl (right) side. The namespace number values (for example, ns1, ns2) can vary.

The Data Mapper dialog should look like the following now.

```
<OrdenPedidoEvento.mplan> (tab)
```

```
<ordenPedidoEvento.mplan> (tab)
```

```
<OrdenPedidoEvento.mplan> (tab)
```

```d.
Select Save All from the File main menu to save your work.
```

e. Click X in the NewOrderSubmitted_To_WarehouseRequest.xsl tab to close the Data Mapper.

With the OrderPendingEvent.mplan tab back in focus, in the Routing Rules section, you should see the transformation updated as follows:

```
<OrdenPedidoEvento.mplan> (tab)
```

```
<ordenPedidoEvento.mplan> (tab)
```

```
<OrdenPedidoEvento.mplan> (tab)
```
Adding a Flow to Update Orders

In this chapter, you provide functionality for updated orders. You create a mediator service component to publish business event `OrderUpdateEvent` and send the order ID for the order to the `OrderProcessor` BPEL process. You create a second mediator service component to subscribe to the business event, send the order ID to the `StoreFrontService` service, and wait for the `StoreFrontService` service to respond with updated order information.

This chapter contains the following sections:

- **Section 8.1, "Task 1: Copy Schema File Needed for Business Event"**
- **Section 8.2, "Task 2: Create the UpdateOrderStatus Mediator Service Component to Publish the OrderUpdateEvent Business Event"**
- **Section 8.3, "Task 3: Create a Routing Rule to Initiate the Business Event"**
- **Section 8.4, "Task 4: Create the OrderUpdateEventMediator Mediator Service Component to Subscribe to the OrderUpdateEvent Business Event"**
- **Section 8.5, "Task 5: Create a Routing Rule to Send Order Updates to the StoreFrontService service"**
- **Section 8.6, "Task 6: Redeploy the OrderBookingComposite Composite"**
- **Section 8.7, "Task 7: Initiate a Test Instance for the OrderBookingComposite Composite"**

### 8.1 Task 1: Copy Schema File Needed for Business Event

To obtain the schema:

1. Copy `OrderProcessor.xsd` from directory `DEMODOWNLOAD_HOME\CompositeServices\OrderBookingComposite\xsd` to `MY_FOD_HOME\CompositeServices\OrderBookingComposite\xsd`. This schema file contains the additional `updateOrderStatus` schema element to create the business event.

2. In the Application Navigator, click the **Refresh** icon.

### 8.2 Task 2: Create the UpdateOrderStatus Mediator Service Component to Publish the OrderUpdateEvent Business Event

To create business event `OrderUpdateEvent` and create a mediator to publish the business event:

...
1. Drag a Mediator service component into the SOA Composite Editor. This service component enables you to subscribe to the business event.

2. In the Name field, enter UpdateOrderStatus.

3. From Template, select Interface Definition from WSDL.

4. Select the Create Composite Service with SOAP Bindings.

5. From the WSDL URL field, generate a WSDL file:
   a. Click the Generate WSDL from schema(s) icon.
   b. From the Create WSDL dialog, in the URL field, click the browse for schema file icon.
   c. In the Type Chooser dialog, expand Project Schema Files > OrderProcessor.xsd and select updateOrderStatus.
   d. Click OK.
   e. Back in the Create WSDL dialog, in the Namespace field, enter http://www.globalcompany.example.com/ns/OrderBookingService.
   f. Click OK and return to the Create Mediator dialog.

6. Back in the Create Mediator dialog, click OK to create the mediator with the settings.

8.3 Task 3: Create a Routing Rule to Initiate the Business Event

To create a routing rule to initiate the business event:

1. Double-click the UpdateOrderStatus mediator.

2. In the Routing Rules section of the Mediator Editor, from the Create dropdown list, select static routing rule to create a routing rule to initiate the business event.
3. In the Target Type message dialog, click **Event**.

4. In the Event Chooser dialog, click the **Create new event definition file (edl)** icon. It is the second icon to the right of the field.

5. In the **Event Definition name** field, enter **OrderEventsDefinition**.

6. In the **Namespace** field, enter http://www.globalcompany.example.com/ns/OrderBookingService.

7. Create business event **OrderUpdateEvent**:
   a. From the **Events** table, click the **Add an Event** icon to add an event.
      The Add an Event dialog appears.
   b. Enter and select the following values:
      
      | Element | Value |
      |---------|-------|
      | Name    | OrderUpdateEvent |
      
      c. Click **OK** to display the Event Definition dialog.
      d. Click **OK** to save settings.
      The Event Chooser dialog displays with the **OrderUpdateEvent** event.
      e. Click **OK** in the Event Chooser dialog.
      The Routing Rules updates with a routing rule to initiate business event **OrderUpdateEvent**.

8. Modify the transformation used for the **UpdateOrderStatus** mediator service components, so that any service subscribing to this event receives input from the **OrderUpdateEvent** business event:
   a. Click the transformation icon next to the **Transform Using** field.
b. In the Request Transformation Map dialog, select Create New Mapper File and leave the default file entry as updateOrderStatus_To_OrderUpdateEvent.xsl, and then click OK.

The Data Mapper displays.

c. On the Source:UpdateOrderStatus.wsdl (left) side, click and drag tns:orderId to orderId on the XSLT File: OrderProcessor.xsd (right) side.

d. On the Source:UpdateOrderStatus.wsdl (left) side, click and drag tns:orderStatus to orderStatus on the XSLT File: OrderProcessor.xsd (right) side. The namespace in your environment may vary.

The Data Mapper dialog should look like the following now.

![Data Mapper dialog](image)

e. Select Save All from the File main menu to save your work.

f. Click X in the updateOrderStatus_To_OrderUpdateEvent.xsl tab to close the Data Mapper.

g. With the UpdateOrderStatus.mplan tab back in focus, in the Routing Rules section, you should see file updateOrderStatus_To_OrderUpdateEvent.xsl in the Transform Using field.

9. Click X in the UpdateOrderStatus.mplan tab to close Mediator Editor.

### 8.4 Task 4: Create the OrderUpdateEventMediator Mediator Service Component to Subscribe to the OrderUpdateEvent Business Event

To create a mediator to subscribe to the OrderUpdateEvent business event and initiate the StoreFrontService service:

1. Drag a Mediator service component into the SOA Composite Editor. This service component enables you to subscribe to the business event.

2. In the Name field, enter OrderUpdateEventMediator.

3. From the Templates list, select Subscribe to Events.

   The window refreshes to display an events table.

4. Click the Subscribe to a new event icon to display the Event Chooser dialog.

5. With the OrderUpdateEvent event selected, click OK.

   You are returned to the Create Mediator dialog.
6. In the Create Mediator dialog, click OK.

The OrderUpdateEventMediator mediator displays in the SOA Composite Editor. The icon on the left side that indicates that mediator is configured for an event subscription.

8.5 Task 5: Create a Routing Rule to Send Order Updates to the StoreFrontService service

To create routing rules to send the order ID to the StoreFrontService service, and wait for the StoreFrontService service to respond with updated order information.

1. Double-click the OrderUpdateEventMediator mediator.

2. In the Routing Rules section of the Mediator Editor, from the Create dropdown list, select static routing rule to create a routing rule to initiate the business event.

3. In the Target Type message dialog, click Service.

4. In the Target Services dialog, expand References > StoreFrontService and select getOrderInfoVOSDO and then click OK.

5. Modify the parameter transformation so that the StoreFrontService service receives the order ID information from the mediator.
a. Click the transformation icon next to the **Transform Using** field.

b. In the Request Transformation Map dialog, select **Create New Mapper File** and leave the default file entry as `OrderUpdateEvent_To_getOrderInfoVOSDO.xsl`, and then click **OK**. The Data Mapper displays.

c. On the **Source**: `OrderProcessor.xsd` (left) side, click and drag `orderId` to `types:orderId` on the **StoreFrontService.wsdl** (right) side.

The Data Mapper dialog should look like the following now.

![Data Mapper dialog](image)

d. Select **Save All** from the **File** main menu to save your work.

e. Click X in the `OrderUpdateEvent_To_getOrderInfoVOSDO.xsl` tab to close the Data Mapper.

f. With the **OrderUpdateEventMediator.mplan** tab back in focus, in the **Routing Rules** section, you should see file `OrderUpdateEvent_To_getOrderInfoVOSDO.xsl` in the **Transform Using** field.

6. In the **Synchronous Reply** section of the Mediator Editor, click the **Browse for target service operations** icon:

![Browse for target service operations](image)

7. In the **Target Type message dialog**, click **Service**.

8. In the Target Services dialog, expand **References** > **StoreFrontService** and select `updateOrderInfoVOSDO` and then click **OK**.

9. Modify the parameter transformation so that the **StoreFrontService** service receives the proper information from the mediator:

a. Click the transformation icon next to the **Transform Using** field.

b. In the Request Transformation Map dialog, select **Create New Mapper File** and leave the default file entry as `getOrderInfoVOSDOResponse_To_updateOrderInfoVOSDO.xsl`, and then click **OK**. The Data Mapper displays.

c. On the **Source** (left) side, click and drag `types:getOrderInfoVOSDOResponse > types:result` to `types:updateOrderInfoVOSDO > types:orderInfoVO1` on the XSLT File right side.

The Auto Map Preferences dialog prompts to perform automatic mapping of the node.

d. Deselect option **Match Elements Considering their Ancestor names**.

e. Leave the default settings for the other options, and then click **OK**.

f. On the **Source**: `StoreFrontService.wsdl` side, expand `types:results > ns4:OrderItemsInfoVO`. 


g. On the XSLT File:StoreFrontService.wsdl side, expand types:orderInfoVO1 > for-each > ns4:OrderItemsInfoVO. The namespace number values (for example, ns1, ns2) can vary.

In the Data Mapper, you can now see how the data is transformed for an updated order.

h. Select Save All from the File main menu to save your work.

i. Click X in the getOrderInfoVOSDOResponse.xsl tab to close the Data Mapper.

j. With the OrderUpdateEventMediator.mplan tab back in focus, in the Routing Rules section, you should see file getOrderInfoVOSDOResponse_To_UpdateOrderInfoVOSDO.xsl in the Transform Using field.

10. Click X in the OrderUpdateEventMediator.mplan tab to close Mediator Editor.

8.6 Task 6: Redeploy the OrderBookingComposite Composite

To redeploy the OrderBookingComposite composite:
1. In the Application Navigator, right-click **OrderBookingComposite** and select Deploy > **OrderBookingComposite** > to **MyAppServerConnection**.
   The SOA Deployment Configuration Dialog displays.

2. Select **Overwrite any existing composite with the same revision ID** to overwrite the composite you previously deployed.

3. When prompted with the Authorization Request dialog, enter **weblogic** in the Username field and the password in the Password field.
   In SOA - Log, a series of validations display, followed by:
   ```
   BUILD SUCCESSFUL
   Total time: nn seconds
   ```

### 8.7 Task 7: Initiate a Test Instance for the OrderBookingComposite Composite

Initiate a test instance of the **OrderBookingComposite** composite, as you have done previously. See Section 5.7.6, "Task 6: Initiate a Test Instance for the OrderBookingComposite Composite" for further information on creating a test instance. For this test instance, make the following adjustments:

- From the **Test** dropdown list, select **UpdateOrderStatus_ep** as the test service.
- In the Inputs Arguments section, enter the following values:
  - In the **orderID** field, enter an ID under 1000.
  - In the **orderStatus** field, enter any value.
Creating the Task Display Form for the ApprovalHumanTask Human Task

In Chapter 5, "Creating the First Half of the OrderProcessor BPEL Process," you created the OrderProcessor BPEL process. The CheckIfRequiresApproval switch of that process uses a human task to pass control to the ApprovalHumanTask human task activity. This human task activity enables a manager named jstein to approve or reject orders totalling more than $2,000. In this chapter, you create the task form for jstein.

This chapter contains the following sections:

- Section 9.1, "About the Task Form"
- Section 9.2, "Task 1: Create a New Task Form for the ApprovalHumanTask Human Task"
- Section 9.3, "Task 2: Add the ADF Business Components Service Runtime Library to the Project"
- Section 9.4, "Task 3: Create the Contents for the Task Form"
- Section 9.5, "Task 4: Deploy the OrderApprovalHumanTask Task Form"

9.1 About the Task Form

The task form is a way for jstein to interact with the ApprovalHumanTask human task. The task form displays the contents of the task to a user’s worklist. Earlier, when you deployed and ran Fusion Order Demo in Chapter 2, you can use the Oracle BPM Worklist to display all the worklist tasks and approve or deny orders totalling more than $2,000.

You create the task form using Oracle Application Development Framework (Oracle ADF) in Oracle JDeveloper. With Oracle ADF, you can design a task display form that depicts the human task in the SOA composite.

The task form is a Java Server Page XML (.jspx) file that you create in a new project of the WebLogicFusionOrderDemo application. Figure 9–1 shows a sample worklist.
Task 1: Create a New Task Form for the ApprovalHumanTask Human Task

When you create a task form based on a human task, Oracle JDeveloper performs the following during task-flow creation:

- Creates data controls based on the task parameters and outcomes
- Creates the initial task form, including the payload

To create a task form based on the ApprovalHumanTask human task:

1. Open the OrderProcessor BPEL process within the SOA composite application.
2. Scroll to the Scope_CheckApprovalLimit scope and expand it.
3. Expand the sequence.
4. Right-click the ApprovalHumanTask_1 human task activity and select Auto-Generate Task Form, as shown in the following figure.

The Create Project dialog displays.

5. Enter OrderApprovalHumanTask for the project name and click OK.

The ApprovalHumanTask_TaskFlow.xml tab displays with the task flow definition and the taskDetails1.jspx tab displays the JSP page with the payload.
9.3 Task 2: Add the ADF Business Components Service Runtime Library to the Project

To add the ADF Business Components service run-time library:

1. In the Application Navigator, right-click OrderApprovalHumanTask and select Project Properties.
2. Select Libraries and Classpath, and from the Libraries and Classpath page, and click Add Library.
3. In the Add Library dialog, select BC4J Service Runtime, and then click OK.
4. In the Libraries and Classpath page, click OK.

9.4 Task 3: Create the Contents for the Task Form

To provide input from the StoreFrontService service in the Contents showDetailHeader for the task form.

1. Add a data control for the StoreFrontService service.
   a. In the Application Navigator, right-click the OrderApprovalHumanTask project and select New.
   b. In the New Gallery dialog, click the All Technologies tab.
   c. In the Categories tree, select Business Tier, and then Data Controls.
   d. Select Web Service Data Control and click OK.
      The Create Web Service Data Control - Step 1 of 5 page displays.
   e. In the Name field, enter StoreFrontService.
   f. In the URL field, click Browse and select StoreFrontService.wsdl in MY_FOD_HOME\CompositeServices\OrderBookingComposite\services\oracle\fodemo\storefront\store\service\common\serviceinterface.

In many ways, this process is similar to creating a references to a service in the SOA Composite Editor.
Task 3: Create the Contents for the Task Form

1. In the Create Web Service Data Control – Step 1 of 5 page, click Next.
2. In the Create Web Service Data Control – Step 2 of 5 page, select the getOrderByInfoVOSDO operation from the Available list and click the Add button. Click Next to proceed to the next page in the wizard.
3. In the Create Web Service Data Control – Step 3 of 5 page, accept the default, and click Finish.

The StoreFrontService data control displays in the Data Controls panel of the Application Navigator.

4. In the Data Controls panel of the Application Navigator, expand the StoreFrontService > getOrderByInfoVOSDO > Return > result to see the data controls you can include in the form.

   ![Data Controls](image)

   - AccountNumber
   - Address1
   - Address2
   - AddressId
   - BillingAddressId
   - CardTypeCode
   - CheckDigits
   - City
   - CountryId
   - CustomerCollectedFlag
   - CustomerId
   - Expirdate
   - FreeShippingFlag
   - GiftCardFlag
   - InstructionText
   - ObjectVersionId
   - ObjectVersionId2
   - ObjectVersionId3
   - OrderDate
   - OrderId

2. In the taskDetails1.jspx page, select the CONTENTS showDetailHeader and from the menu, select Design this Container.

3. With the CONTENTS showDetailHeader still selected, from the menu, select Disclose Show Detail Header.

   ![Design This Container](image)

   The CONTENTS header shows the order ID is included.

   ![Contents](image)

   4. Select the panelFormLayout containing the orderID input label and click Enter.

   The following image shows the panelgroupLayout being selected.

   ![Panel Group Layout](image)
Task 3: Create the Contents for the Task Form

Creating the Task Display Form for the ApprovalHumanTask Human Task

After you click Enter, a new label displays.

5. Add parameters from the StoreFrontService service into the task form:
   a. In the Data Controls panel of the Application Navigator, drag the result icon into the empty input field.
   b. From the Create menu, select Forms, and then ADF Read-only Form, as shown in the following figure.

   ![Edit Form Fields dialog](image)

   The Edit Action dialog displays with the ApprovalHumanTask_ApprovalHumanTask data control selected.
   c. In the Edit Form Fields dialog, select all but the following fields and click the Delete icon:
      - OrderStatusCode
      - OrderTotal
      - ShipToName
      - ShipToPhoneNumber
      - Address1
      - Address2
      - City
      - PostalCode
      - StateProvince
      - CountyId

   The Edit Form Fields dialog should look like the following image:
d. Click **OK**.

The Edit Action dialog displays with the **StoreFrontService** data collection selected.

e. In the **Parameters** section, in the **Value** field, select **Show El Expression Builder**.

The Variables dialog displays.

f. Expand **ADF Bindings > bindings** and select **orderId** and then select **OK**.
g. In the Edit Action dialog, click OK.

   The CONTENTS header shows the selected input parameters:

6. Select Save All from the File main menu to save your work.

9.5 Task 4: Deploy the OrderApprovalHumanTask Task Form

To deploy the OrderApprovalHumanTask form:

1. In the Application Navigator, right-click OrderApprovalHumanTask and select Deploy > OrderApprovalHumanTask > to MyAppServerConnection.
   
   The SOA Deployment Configuration Dialog displays.
2. Accept the default settings and click **OK**.

3. When prompted with the Authorization Request dialog, enter `weblogic` in the **Username** field and the password in the **Password** field.

   In SOA - Log, a series of validations display, followed by:

   ```
   BUILD SUCCESSFUL
   Total time: nn seconds
   ```
Congratulations! You created an SOA composite application using the following components:

- Oracle Mediator
- Oracle BPEL Process Manager
- Human Oracle Human Workflow (using a human task)
- Oracle Business Rules
- Oracle Messaging Service

You should now have a working knowledge of the advantages provided by Oracle SOA Suite. To learn more, refer to the following resources:

- Oracle Fusion Middleware Getting Started with Oracle SOA Suite provides an overview of Oracle SOA Suite and other Oracle products that complement your SOA environment.
- Oracle Fusion Middleware Administrator’s Guide for Oracle SOA Suite describes how to administer the run-time environment for Oracle SOA Suite, with a focus on Fusion Middleware Control.
- Oracle SOA Suite on Oracle Technology Network provides access to various use-case samples for Oracle SOA Suite and its components.

http://www.oracle.com/technology/sample_code/products/soa
Create the JMS Topic for the FulfillmentBatch Adapter

This appendix complements Section 6.4, "Creating the Services and Routing Required for the Scope_FulfillOrder Scope." It contains the following tasks:

- Section A.1, "Task 1: Create the JMS Topic"
- Section A.2, "Task 2: Create the JMS Topic Connection Factory"
- Section A.3, "Task 3: Add the Connection Pool"

A.1 Task 1: Create the JMS Topic

The DemoSupplierTopic topic defines a publish and subscribe destination type, which is used for asynchronous peer communications.

To create it through the Oracle WebLogic Server Administration Console.

1. Access the Oracle WebLogic Server Console from the following URL:
   
   http://hostname:port/console
   
   where is the hostname is the DNS name or IP address of the Administration Server for Oracle SOA Suite and port is the DNS name or IP address of the Administration Server and port is the address of the port on which the Administration Server is listening for requests (7001 by default).

2. When the login page appears, enter the user name and the password you used to start the Administration Server (you may have specified this user name and password during the installation process), or enter a user name that is granted a default global security role.

3. Click Log In.

4. In the Administration Console, from the navigation pane, expand Services > Messaging and select JMS Modules.

5. On the JMS Modules page, select SOAJMSModule.


7. On the Create a New JMS System Module Resource page, select Topic and then Next.

8. In the JMS Destination Properties section, enter the following details:
Task 2: Create the JMS Topic Connection Factory

To add the connection factory, which defines a set of connection configuration parameters to create connections for JMS clients:

1. On the Settings for SOAJMSModule page, in the Summary of Resources table, select New again.
2. On the Create a New JMS System Module Resource page, select Connection Factory and then Next.
3. In the Connection Factory Properties section, enter the following details:

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>DemoSupplierTopicCF</td>
</tr>
<tr>
<td>JNDI Name</td>
<td>jms/DemoSupplierTopicCF</td>
</tr>
</tbody>
</table>

9. Click Next.

10. From the Subdeployments list, select SOASubDeployment.

11. From the JMS Servers table, select SOAJMSServer, and then click Finish.

The Summary of Resources table now displays the topic DemoSupplierTopic.

A.2 Task 2: Create the JMS Topic Connection Factory

The following properties will be used to identify your new Topic. The current module is SOAJMSModule.

* Indicates required fields

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>DemoSupplierTopic</td>
</tr>
<tr>
<td>JNDI Name</td>
<td>jms/DemoSupplierTopic</td>
</tr>
<tr>
<td>Name</td>
<td>DemoSupplierTopicCF</td>
</tr>
<tr>
<td>JNDI Name</td>
<td>jms/DemoSupplierTopicCF</td>
</tr>
</tbody>
</table>
A.3 Task 3: Add the Connection Pool

A connection pool is configured in the JMSAdapter application.

To add a connection pool:

1. The JMSAdapter application uses a deployment plan. Create a directory for deployment plan in the following directory:
   - (UNIX) $MW_HOME/soa/JSMPlan
   - (Windows) $MW_HOME\soa\JSMPlan

2. In the Administration Console, from the navigation pane, select Deployments.

3. On the Summary of Deployments page, from the Deployments table, select the JmsAdapter link.

4. On the Settings for JmsAdapter page, select the Configuration tab and then the Outbound Connection Pools sub-tab.

5. From the Outbound Connection Pool Configuration table, click New.

6. From the Outbound Connection Groups table, select oracle.tip.adapter.jms.IJmsConnectionFactory, and then click Next.

7. In the JNDI Name field, enter eis/Jms/TopicConnectionFactory.

8. Click Finish.
9. When prompted to select a deployment plan, perform the following steps:
   a. In the Path field, select the path to the directory you created in Step 1 and enter Plan.xml after the path.
      If there is a plan file selected with the option at the top of the page, the one you enter takes precedence.
   b. Click OK.
   c. Verify the plan name is set to Plan.xml in the JMSPlan directory.

10. Select the Configuration tab.

11. Expand the select oracle.tip.adapter.jms.IJmsConnectionFactory, and select eis/Jms/TopicConnectionFactory

12. For the following rows, select the Property Value cell on the far right, modify the value as specified, and then press Enter.

<table>
<thead>
<tr>
<th>Row</th>
<th>Property Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConnectionFactoryLocation</td>
<td>jms/DemoSupplierTopicCF</td>
</tr>
<tr>
<td>IsTopic</td>
<td>true</td>
</tr>
<tr>
<td>Username</td>
<td>The Oracle WebLogic Server administrator. For example: weblogic</td>
</tr>
<tr>
<td>Password</td>
<td>The password of the Oracle WebLogic Server weblogic user. For example: welcome1</td>
</tr>
</tbody>
</table>

13. Click Save.

14. Redeploy the adapter:
   a. From the navigation pane, select Deployments.
      The Summary of Deployments page displays with the Deployments table.
   b. Select the check box next to the JmsAdapter, and then click Update.
   c. In the Update Application Assistant page, click Redeploy this application using the following deployment file.
   d. Click Next.
   e. Click Finish.
This appendix contains the following sections:

- Section B.1, "About ant Scripts"
- Section B.2, "ant Targets for WebLogicFusionOrderDemo"

### B.1 About ant Scripts

ant is a Java-based build tool used by Oracle SOA Suite for managing SOA composite applications. The WebLogic Fusion Order Demo application provides an example of using ant scripts to compile, package, and deploy the application.

The following files control the ant scripts:

- `build.properties` in `DEMO_DOWNLOAD_HOME/CompositeServices/bin`:
  A file that you edit to reflect your environment (for example, specifying Oracle home and Java home directories, setting server properties such as host name and port number to use for deployment, specifying the application to deploy, and so on).

- `build.xml` in `DEMO_DOWNLOAD_HOME/CompositeServices/bin`:
  Used by ant to compile, build, and deploy composite applications to the server specified in the `build.properties` file.

To use ant scripts for specified ant targets:

1. Modify the `build.properties` file to reflect your environment.
2. In the Application Navigator, under the Resources node, right-click `build.xml` and choose Run Ant Target and select appropriate ant target.
   This command builds the targets defined in the `build.xml` file.

### B.2 ant Targets for WebLogicFusionOrderDemo

Table B–1 describes the targets available in the `build.xml` file. It shows which other targets are dependent or called from a target. If a called target exists in another file, the table references the file and location as directory > file_name > target.)
<table>
<thead>
<tr>
<th>ant Target</th>
<th>Description</th>
<th>Dependent Targets and Targets Called From Other Targets. Targets from another file display as directory &gt; file_name &gt; target.</th>
</tr>
</thead>
<tbody>
<tr>
<td>build.src.zip</td>
<td>This script creates the source distribution.</td>
<td>Depends on clean</td>
</tr>
<tr>
<td>clean</td>
<td>This script removes existing sources.</td>
<td>Depends on init and runs the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1. orderbooking.composite.home/bin &gt; build-sca-composite.xml &gt; clean</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2. orderbooking.bam.home/bin &gt; build-sca-composite.xml &gt; clean</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3. partnersupplier.composite.home/bin &gt; build-sca-composite.xml &gt; clean</td>
</tr>
<tr>
<td>compile-build-all</td>
<td>This script compiles and builds all applicable composites and applications.</td>
<td>Depends on init and runs the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4. b2b.composite.home/bin &gt; build-sca-composite.xml &gt; clean</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5. orderbookingsdo.composite.home/bin &gt; build-sca-composite.xml &gt; clean</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6. creditauthorization.home/bin &gt; build-sca-composite.xml &gt; clean</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7. creditauthorization.home/bin &gt; build-sca-composite.xml &gt; clean</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8. orderapproval.home/bin/bin &gt; build-sca-composite.xml &gt; clean</td>
</tr>
</tbody>
</table>
## Table B–1 (Cont.) ant Targets for WebLogicFusionOrderDemo

<table>
<thead>
<tr>
<th>ant Target</th>
<th>Description</th>
<th>Dependent Targets and Targets Called From Other Targets. Targets from another file display as directory &gt; file_name &gt; target.</th>
</tr>
</thead>
<tbody>
<tr>
<td>compile-deploy-all</td>
<td>This script compiles, builds, and deploys all applicable composites and applications.</td>
<td>Depends on compile-build-all and runs the following: 1. orderbooking.composite.home/bin &gt; build-sca-composite.xml &gt; deploy-composite -ORACLE_HOME/bin &gt; ant-sca-compile.xml &gt; attachplan -ORACLE_HOME/bin &gt; ant-sca-deploy.xml &gt; deploy 2. orderbooking.bam.home/bin &gt; build-sca-composite.xml &gt; deploy-composite -ORACLE_HOME/bin &gt; ant-sca-compile.xml &gt; attachplan -ORACLE_HOME/bin &gt; ant-sca-deploy.xml &gt; deploy 3. partnersupplier.composite.home/bin &gt; build-sca-composite.xml &gt; deploy-composite -ORACLE_HOME/bin &gt; ant-sca-compile.xml &gt; attachplan 4. b2b.composite.home/bin &gt; build-sca-composite.xml &gt; deploy-composite -ORACLE_HOME/bin &gt; ant-sca-compile.xml &gt; attachplan -ORACLE_HOME/bin &gt; ant-sca-deploy.xml &gt; deploy 5. orderbookingsdo.composite.home/bin &gt; build-sca-composite.xml &gt; deploy-composite -ORACLE_HOME/bin &gt; ant-sca-compile.xml &gt; attachplan 6. creditauthorization.home/bin &gt; build.xml &gt; deploy-application 7. orderapproval.home/bin &gt; build.xml &gt; deploy-application</td>
</tr>
<tr>
<td>createMDSConnections</td>
<td>This script seeds Oracle Metadata Repository connection information.</td>
<td>Runs the following: 1. createMDSConnectionsForDB 2. createMDSConnectionsForFileStore 3. createMDSConnectionsForServerFileStore</td>
</tr>
<tr>
<td>createMDSConnectionsForDB</td>
<td>This script seeds the connections for a database-based Oracle Metadata Repository.</td>
<td>build.properties &gt; foreign.mds.type=db</td>
</tr>
<tr>
<td>createMDSConnectionsForFileStore</td>
<td>This script seeds the connections for a file-based Oracle Metadata Repository.</td>
<td>build.properties &gt; foreign.mds.type=jdev</td>
</tr>
<tr>
<td>createMDSConnectionsForServerFileStore</td>
<td>This script seeds the connections for a database-based Oracle Metadata Repository.</td>
<td>build.properties &gt; foreign.mds.type=server.file</td>
</tr>
<tr>
<td>init</td>
<td>This script displays build information.</td>
<td>Not applicable</td>
</tr>
</tbody>
</table>

ant Targets for WebLogicFusionOrderDemo

ant Scripts  B-3
### Table B–1  (Cont.) ant Targets for WebLogicFusionOrderDemo

<table>
<thead>
<tr>
<th>ant Target</th>
<th>Description</th>
<th>Dependent Targets and Targets Called From Other Targets. Targets from another file display as directory &gt; file_name &gt; target.</th>
</tr>
</thead>
<tbody>
<tr>
<td>jdeveloper-setup-seed</td>
<td>This script complete client-side setup.</td>
<td>Depends on init and runs the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1. createMDSConnections</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2. seedFodJmsResources</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3. setupWorkspaceForJdeveloperUse</td>
</tr>
<tr>
<td>removeDemoUsers</td>
<td>This script removes the demo community</td>
<td>Runs deploy SOATestDemoApp -action REMOVE.COMMUNITY</td>
</tr>
<tr>
<td>removeFodJmsResources</td>
<td>This script removes the JMS resources.</td>
<td>Not applicable</td>
</tr>
<tr>
<td>seedB2BAgreements</td>
<td>This script seeds Trading Partner agreements.</td>
<td>b2b.composite.home/bin &gt; build-sca-composite.xml &gt; importAndDeployB2BtradingAgreements</td>
</tr>
<tr>
<td>seedDemoUsers</td>
<td>This script seeds the demo community. For this tutorial, it adds jstein as an user to approve orders for over $2,000. When you run the demo, you place an order for $2,000 and log in to the Oracle BPM Worklist as jstein and approve the order.</td>
<td>Runs deploy SOATestDemoApp -action SEED.COMMUNITY SOATestDemoApp is a server-side application to seed the users.</td>
</tr>
<tr>
<td>seedFodJmsResources</td>
<td>This script creates the needed JMS resources.</td>
<td>Not applicable</td>
</tr>
</tbody>
</table>
### Table B-1 (Cont.) ant Targets for WebLogicFusionOrderDemo

<table>
<thead>
<tr>
<th>ant Target</th>
<th>Description</th>
<th>Dependent Targets and Targets Called From Other Targets. Targets from another file display as directory &gt; file_name &gt; target.</th>
</tr>
</thead>
<tbody>
<tr>
<td>setupWorkspaceForJDeveloperUse</td>
<td>This script sets up the application workspace in Oracle JDeveloper.</td>
<td>Runs the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1. orderbooking.composite.home/bin &gt; build-sca-composite.xml &gt; setupDeploymentEnvironment</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2. orderbooking.bam.home/bin &gt; build-sca-composite.xml &gt; setupDeploymentEnvironment</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3. b2b.composite.home/bin &gt; build-sca-composite.xml &gt; deploy-composite</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4. createMDSConnection</td>
</tr>
<tr>
<td>server-cleanup-all</td>
<td>This script removes existing sources and undeploys applications.</td>
<td>Depends on init and runs the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1. orderbooking.composite.home/bin &gt; build-sca-composite.xml &gt; undeploy-composite</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-ORACLE_HOME/bin &gt; ant-sca-test.xml &gt; undeploy</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2. orderbooking.bam.home/bin &gt; build-sca-composite.xml &gt; undeploy-composite</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-ORACLE_HOME/bin &gt; ant-sca-test.xml &gt; undeploy</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3. partnersupplier.composite.home/bin &gt; build-sca-composite.xml &gt; undeploy-composite</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-ORACLE_HOME/bin &gt; ant-sca-test.xml &gt; undeploy</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4. b2b.composite.home/bin &gt; build-sca-composite.xml &gt; undeploy-composite</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-ORACLE_HOME/bin &gt; ant-sca-test.xml &gt; undeploy</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5. orderbookingsdo.composite.home/bin &gt; build-sca-composite.xml &gt; undeploy-composite</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-ORACLE_HOME/bin &gt; ant-sca-test.xml &gt; undeploy</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6. creditauthorization.home/bin &gt; build.xml &gt; undeploy-application</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7. orderapproval.home/bin &gt; build.xml &gt; undeploy-application</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-b2b.composite.home/bin &gt; build-sca-composite.xml &gt; purgeB2BTradingAgreements</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8. removeFodJmsResources</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9. removeDemoUsers</td>
</tr>
</tbody>
</table>
### Table B–1 (Cont.) ant Targets for WebLogicFusionOrderDemo

<table>
<thead>
<tr>
<th>ant Target</th>
<th>Description</th>
<th>Dependent Targets and Targets Called From Other Targets. Targets from another file display as <code>directory &gt; file_name &gt; target</code>.</th>
</tr>
</thead>
</table>
| server-setup-seed-deploy-test | This script runs all the other targets. | Runs the following:  
1. createMDSConnections  
2. compile-deploy-all  
3. seedPodJmsResources  
4. seedB2BAgreements  
5. seedDemoUsers  
6. orderbooking.bam.home/bin>build-sca-composite.xml>seedBAMServerObjects  
   Depends on createBAMConfig  
7. orderbooking.bam.home/bin>build-sca-composite.xml>seedBAMAdapterResources  
8. orderbooking.composite.home>build-sca-composite.xml>test-composite  
   `-ORACLE_HOME/bin>ant-sca-test.xml@test `  
9. orderbooking.bam.home>build-sca-composite.xml>test-composite  
   `-ORACLE_HOME/bin>ant-sca-test.xml@test` |
A
admin.server.host parameter, 2-4
admin.server.port parameter, 2-4
ant scripts
  build.properties file, B-1
  build.src.zip, B-2
  build.xml file, B-1
  clean, B-2
  compile-build-all, 2-5, B-2
  compile-deploy-all, 2-6, B-3
  createMDSConnections, B-3
  createMDSConnectionsForDB, B-3
  createMDSConnectionsForFileStore, B-3
  createMDSConnectionsForServerFileStore, B-3
described, B-1
init, B-3
jdeveloper-setup-seed, B-4
removeDemoUsers, B-4
removeFodJmsResources, B-4
seedB2BAgreements, B-4
seedDemoUsers, 2-5, B-4
seedFodJmsResources, 2-5, B-4
server-cleanup-all, B-5
server-setup-seed-deploy-test, B-6
setupWorkspaceForJDeveloperUse, 2-5, B-5
ApprovalHumanTask human task service
  component, 5-45
  creating a task form from, 9-2
  described, 4-3
assign activity, 3-6

B
B2BX12OrderGateway project, 1-7
BamOrderBookingComposite project, 1-7
BC4J Service Runtime library
  adding to OrderApprovalHumanTask project, 9-3
  adding to OrderBookingComposite project, 4-8
bin project, 1-7
bind entity activity, 5-10
bottom-up approach, 4-4
BPEL Designer
  creating a partner link, 5-17
  creating variables, 5-6
build.properties file, B-1
build.src.zip, B-2
build.xml file, B-1
business events
  NewOrderSubmitted, 7-1
  OrderUpdateEvent, 8-1
business rules
  described, 5-33
  EvaluatePreferredSupplierRule, 6-26
  OrderBookingComposite, used in, 1-10
  RequiresApprovalRule, 5-33

catch branches, 5-29, 6-47
clean ant script, B-2
compile-build-all ant script, 2-5, B-2
compile-deploy-all ant script, 2-6, B-3
createMDSConnections ant script, B-3
createMDSConnectionsForDB ant script, B-3
createMDSConnectionsForFileStore ant script, B-3
createMDSConnectionsForServerFileStore ant script, B-3
CreditCardAuthorization project, 1-7
CreditCardAuthorization Web service, 4-4
CreditCardAuthorizationService.wsdl, 5-17
db.adminUser parameter, 2-2
db.demoUser.tablespace parameter, 2-2
DemoSupplierTopic topic, A-1
deploying
  OrderApprovalHumanTask task form, 9-7
  OrderBookingComposite composite, 5-15
  OrderSDOComposite composite, 5-16
  PartnerSupplierComposite composite, 3-9
e-mail activity, 6-45
entity variables, 5-6
EvaluatePreferredSupplierRule business rule service component, 4-3
creating, 6-28
ExternalPartnerSupplier BPEL process, 3-3
<table>
<thead>
<tr>
<th><strong>F</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>ExternalPartnerSupplier.wsdl file, 3-4, 6-16</td>
</tr>
<tr>
<td>flow activity, 6-6</td>
</tr>
<tr>
<td>pattern, 6-2</td>
</tr>
<tr>
<td>foreign.mds.type parameter, 2-5</td>
</tr>
<tr>
<td>FulfillmentBatch JMS adapter, 4-4</td>
</tr>
<tr>
<td>creating, 6-35</td>
</tr>
<tr>
<td>FulfillmentBatch.jms.jca file, 6-36</td>
</tr>
<tr>
<td>FulfillmentBatch.wsdl file, 6-36, 6-39</td>
</tr>
<tr>
<td>FulfillOrder mediator service component, 4-3</td>
</tr>
<tr>
<td>FulfillOrderRef.wsdl file, 6-39</td>
</tr>
<tr>
<td>Fusion Middleware Control</td>
</tr>
<tr>
<td>monitoring orders, 2-10</td>
</tr>
<tr>
<td>starting, 2-10</td>
</tr>
<tr>
<td>Fusion Order Demo</td>
</tr>
<tr>
<td>installing schema, 2-1</td>
</tr>
<tr>
<td>introduced, 1-1</td>
</tr>
<tr>
<td>running, 2-1 to 2-23</td>
</tr>
<tr>
<td>FusionOrderDemo_R1.zip, 1-2</td>
</tr>
<tr>
<td><strong>G</strong></td>
</tr>
<tr>
<td><strong>H</strong></td>
</tr>
<tr>
<td>human task</td>
</tr>
<tr>
<td>creating a task form from a, 9-2</td>
</tr>
<tr>
<td>creating from BPEL process, 5-45</td>
</tr>
<tr>
<td>human task activity, 5-45</td>
</tr>
<tr>
<td>Human Task Editor</td>
</tr>
<tr>
<td>assigning task participant, 5-47</td>
</tr>
<tr>
<td>defining a title, 5-46</td>
</tr>
<tr>
<td><strong>I</strong></td>
</tr>
<tr>
<td>init ant script, B-3</td>
</tr>
<tr>
<td>installing</td>
</tr>
<tr>
<td>schema for Fusion Order Demo, 2-1</td>
</tr>
<tr>
<td>InternalWarehouseService process, 4-3</td>
</tr>
<tr>
<td>InternalWarehouseService.wsdl file, 6-19</td>
</tr>
<tr>
<td>InternalWarehouse.xsd file, 4-6, 6-7</td>
</tr>
<tr>
<td>invoke activities</td>
</tr>
<tr>
<td>creating, 5-12</td>
</tr>
<tr>
<td>invoke activity, 5-12</td>
</tr>
<tr>
<td><strong>J</strong></td>
</tr>
<tr>
<td>jdbc.port parameter, 2-2</td>
</tr>
<tr>
<td>jdbc.sid parameter, 2-2</td>
</tr>
<tr>
<td>jdbc.urlBase parameter, 2-2</td>
</tr>
<tr>
<td>jdeveloper.home parameter, 2-2</td>
</tr>
<tr>
<td>jdeveloper-setup-seed ant script, B-4</td>
</tr>
<tr>
<td>JMS adapter</td>
</tr>
<tr>
<td>deployment plan, A-3</td>
</tr>
<tr>
<td>JMS connection factory, A-2</td>
</tr>
<tr>
<td>JMS connection pool, A-3</td>
</tr>
<tr>
<td>JMS topic, A-1</td>
</tr>
<tr>
<td>JMS connection factory, A-2</td>
</tr>
<tr>
<td>JMS connection pool, A-3</td>
</tr>
<tr>
<td>JMS topic, A-1</td>
</tr>
<tr>
<td>JMSAdapter application, A-3</td>
</tr>
<tr>
<td><strong>L</strong></td>
</tr>
<tr>
<td>LegacyOrderBookingPO.xsd file, 6-34</td>
</tr>
<tr>
<td><strong>M</strong></td>
</tr>
<tr>
<td>managed.server parameter, 2-4</td>
</tr>
<tr>
<td>managed.server.port parameter, 2-5</td>
</tr>
<tr>
<td>Mediator Editor</td>
</tr>
<tr>
<td>routing rules, 6-15, 7-4, 8-3, 8-5</td>
</tr>
<tr>
<td><strong>N</strong></td>
</tr>
<tr>
<td>NewOrderSubmitted business event</td>
</tr>
<tr>
<td>creating, 7-1</td>
</tr>
<tr>
<td>described, 7-1</td>
</tr>
<tr>
<td>NotificationService Web service, 4-4</td>
</tr>
<tr>
<td><strong>O</strong></td>
</tr>
<tr>
<td>Oracle BPM Worklist</td>
</tr>
<tr>
<td>approving orders, 2-21</td>
</tr>
<tr>
<td>creating a task form, 9-1</td>
</tr>
<tr>
<td>starting, 2-21</td>
</tr>
<tr>
<td>oracle.home parameter, 2-4</td>
</tr>
<tr>
<td>OrderApprovalHumanTask project</td>
</tr>
<tr>
<td>BC4J Service Runtime library, 9-3</td>
</tr>
<tr>
<td>creating, 9-2</td>
</tr>
<tr>
<td>described, 1-7</td>
</tr>
<tr>
<td>OrderApprovalHumanTask task form</td>
</tr>
<tr>
<td>creating contents of, 9-3</td>
</tr>
<tr>
<td>deploying, 9-7</td>
</tr>
<tr>
<td>OrderBookingComposite composite</td>
</tr>
<tr>
<td>ApprovalHumanTask service component, 4-3</td>
</tr>
<tr>
<td>bottom-up approach, 4-4</td>
</tr>
<tr>
<td>business rules, used in, 1-10</td>
</tr>
<tr>
<td>creating, 4-5</td>
</tr>
<tr>
<td>CreditCardAuthorization Web service, 4-4</td>
</tr>
<tr>
<td>deploying, 5-15</td>
</tr>
<tr>
<td>EvaluatePreferredSupplierRule service component, 4-3</td>
</tr>
<tr>
<td>FulfillmentBatch JMS adapter, 4-4</td>
</tr>
<tr>
<td>FulfillOrder service component, 4-3</td>
</tr>
<tr>
<td>InternalWarehouseService service component, 4-3</td>
</tr>
<tr>
<td>NotificationService Web service, 4-4</td>
</tr>
<tr>
<td>OrderPendingEvent service component, 4-3</td>
</tr>
<tr>
<td>OrderProcessor service component, 4-3</td>
</tr>
<tr>
<td>orderprocessor_client_ep service, 4-3</td>
</tr>
<tr>
<td>OrderUpdateEventMediator service component, 4-3</td>
</tr>
<tr>
<td>PartnerSupplierMediator service component, 4-4</td>
</tr>
<tr>
<td>PartnerSupplierService Web service, 4-4</td>
</tr>
<tr>
<td>RequiresApprovalRule service component, 4-4</td>
</tr>
<tr>
<td>StoreFrontService, 4-4</td>
</tr>
<tr>
<td>test instance for, 5-16</td>
</tr>
<tr>
<td>top-down approach, 4-4</td>
</tr>
<tr>
<td>undeploying, 2-23</td>
</tr>
<tr>
<td>UpdateOrderStatus service component, 4-4</td>
</tr>
<tr>
<td>UpdateOrderStatus_ep, 4-3</td>
</tr>
<tr>
<td>USPSShipment file adapter, 4-4</td>
</tr>
</tbody>
</table>
OrderBookingComposite project, 1-7
   BC4J Service Runtime library, 4-8
flow described, 1-8
OrderBookingRules.xsd file, 5-37
OrderEO.edl file, 7-2
OrderEO.xsd file, 7-2
OrderInfoVOSDO.xsd file, 5-7
OrderPendingEvent mediator service component, 4-3
   creating, 7-2
   routing rule for, 7-3
OrderProcessor BPEL process introduced, 4-3
   Scope_AuthorizeCreditCard scope creating, 5-19
described, 4-9
   Scope_CheckApprovalLimit scope creating, 5-35
described, 4-9
   Scope_FulfillOrder scope creating, 6-41
described, 4-9
   Scope_NotifyCustomerofCompletion scope creating, 6-45
described, 4-9
   Scope_RetrieveCustomerForOrder scope creating, 5-12
described, 4-9
   Scope_RetrieveOrder scope creating, 5-9
described, 4-9
   Scope_RetrieveQuotes scope creating, 6-6
described, 4-9
   Scope_SelectPreferredSupplier scope creating, 6-27
described, 4-9
   Scope_UpdateStatusToComplete scope creating, 6-44
described, 4-9
orderprocessor_client_ep service, 4-3
OrderProcessor.wsdl file, 5-25, 7-5
OrderProcessor.xsd file, 8-1
OrderSDOComposite composite deploying, 5-16
OrderSDOComposite project, 1-7
OrderUpdateEvent business event, 8-1
   creating, 8-3
described, 8-1
OrderUpdateEventMediator mediator service component, 4-3
   creating, 8-4
   routing rule for, 8-5

P
PartnerSupplierComposite composite
   creating, 3-3
deploying, 3-9
described, 3-1
test instance for, 3-9
undeploing, 2-23
PartnerSupplierComposite project, 1-7
PartnerSupplierMediator mediator service component
   creating, 4-4, 6-12
PartnerSupplierMediator.wsdl file, 6-16
PartnerSupplierService Web service, 4-4

R
Readme.txt file, 2-6
receive activities, 6-25
removeDemoUsers ant script, B-4
removeFodJmsResources ant script, B-4
RequiresApprovalRule service component
   creating, 5-37
described, 4-4

S
Scope_AuthorizeCreditCard scope
   creating, 5-19
described, 4-9
Scope_CheckApprovalLimit scope
   creating, 5-35
described, 4-9
Scope_FulfillOrder scope
   creating, 6-41
described, 4-9
Scope_NotifyCustomerofCompletion
   described, 4-9
Scope_NotifyCustomerofCompletion scope
   creating, 6-45
Scope_RetrieveCustomerForOrder scope
   creating, 5-12
described, 4-9
Scope_RetrieveOrder scope
   creating, 5-9
described, 4-9
Scope_RetrieveQuotes scope
   creating, 6-6
described, 4-9
Scope_SelectPreferredSupplier scope
   creating, 6-27
described, 4-9
Scope_UpdateStatusToComplete scope
   creating, 6-44
described, 4-9
seedB2BAgreements ant script, B-4
seedDemoUsers ant script, 2-5, B-4
seedFodJmsResources ant script, 2-5, B-4
server-cleanup-all ant script, B-5
server.password parameter, 2-5
server-setup-seed-deploy-test ant script, B-6
server.targets parameter, 2-5
server.user parameter, 2-5
setDomainEnv.cmd file, 1-3
setDomainEnv.sh file, 1-3
setupWorkspaceForJDeveloperUse ant script, 2-5, B-5
soa.only.deployment parameter, 2-4
soa.server.oracle.home parameter, 2-5
StoreFront module
   deploying, 2-2
described, 1-1
   placing orders, 2-6
   StoreFrontService project, 1-1
   StoreFrontUI project, 1-1
   StoreFrontServiceRef.wsdl file, 5-4
   StoreFrontService project, 1-1
   StoreFrontService service
      in OrderBookingComposite, 4-4
      StoreFrontServiceRef.wsdl file, 5-3
      StoreFrontService.wsdl file, 5-16, 8-6
      StoreFrontUI project, 1-1
      switch activity, 5-23
test instances
  OrderBookingComposite, 5-16
  PartnerSupplierComposite, 3-9
throw activity, 5-24
top-down approach, 4-4

undeploying composite applications, 2-23
UpdateOrderStatus mediator service
  component, 4-4
  creating, 8-2
  routing rule for, 8-2
UpdateOrderStatus_ep service, 4-3
UpdateOrderStatus.wsdl file, 8-4
USPShipment_file.jca file, 6-35
USPSShipment file adapter, 4-4
USPSShipment.wsdl file, 6-35

variables
  creating in BPEL Designer, 5-6
  entity, 5-6
  global, 5-6
  local, 5-6

Warehouse.xsd schema file, 3-4
WebLogic Fusion Order Demo application
  B2BX12OrderGateway project, 1-7
  BamOrderBookingComposite project, 1-7
  bin project, 1-7
  creating, 3-2
  CreditCardAuthorization project, 1-7
  deploying, 2-4
  introduced, 1-1
  OrderApprovalHumanTask project, 1-7
  OrderBookingComposite project, 1-7
  OrderSDOComposite project, 1-7
  PartnerSupplierComposite project, 1-7
  setting up, 1-2
  viewing in Oracle JDeveloper, 1-6
WSDL Editor, 5-25