Oracle® Fusion Middleware

Understanding Oracle SOA Suite

14*c* (14.1.2.0.0)
F80345-01
November 2024

Oracle Fusion Middleware Understanding Oracle SOA Suite, 14*c* (14.1.2.0.0)

F80345-01

Primary Author: Oracle Corporation

Contents

10 Gaining Business Insights with Oracle Business Activity Monitoring

Preface

Understanding Oracle SOA Suite provides a high-level introduction to Oracle SOA Suite concepts and components, highlighting business challenges that can be addressed by Oracle SOA Suite components.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at https://www.oracle.com/corporate/accessibility/.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit https://support.oracle.com/portal/ or visit Oracle Accessibility Learning and Support if you are hearing impaired.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

Related Documents

Refer to the Oracle Fusion Middleware library on the Oracle Help Center for additional information.

- For Oracle SOA Suite information, see Oracle SOA Suite.
- For adapters information, see On-Premises and Cloud SOA Adapters.
- For Oracle BAM information, see Oracle Business Activity Monitoring.
- For Oracle B2B information, see Oracle B2B.
- For Oracle Business Process Management information, see Oracle Business Process Management.
- For Oracle Enterprise Scheduler information, see Oracle Enterprise Scheduler.
- For Oracle Managed File Transfer information, see Oracle Managed File Transfer.
- For Oracle Service Bus information, see Oracle Service Bus.

- For Oracle SOA Suite for healthcare integration information, see Oracle SOA Suite for Healthcare Integration.
- For versions of platforms and related software for which Oracle products are certified and supported, review the Certification Matrix on OTN.
- For cloud adapters information, see :
  - *Using the Ariba Adapter*
  - *Using the Oracle Eloqua Cloud Adapter*
  - *Using the Oracle ERP Cloud Adapter*
  - *Using the Oracle NetSuite Adapter*
  - *Using the Oracle RightNow Adapter*
  - *Using the Salesforce Adapter*
  - *Using the Oracle Sales Cloud Adapter*
  - *Using the ServiceNow Adapter*
  - *Oracle Cloud Adapters Postinstallation Configuration Guide*

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Part I
Introduction to Oracle SOA Suite

This part includes the following chapter:

1

Overview of Oracle SOA Suite

This chapter provides a high-level overview of Oracle SOA Suite architecture, including key concepts, components, management tools and processes, and process flows.
For more information about:

- Oracle SOA Suite architecture and building applications, see *Developing SOA Applications with Oracle SOA Suite*
- Oracle SOA Suite infrastructure and administration, see *Administering Oracle SOA Suite and Oracle Business Process Management Suite*

This chapter includes the following sections:

About Oracle SOA Suite

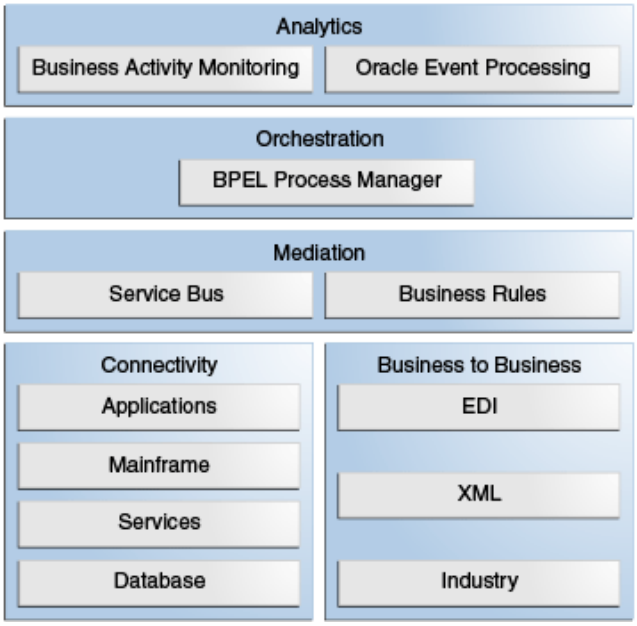
Oracle SOA Suite is a comprehensive, hot-pluggable software suite that enables you to build, deploy, and manage integrations using service-oriented architecture (SOA). Oracle SOA Suite provide the following capabilities:

- Consistent tooling
- A single deployment and management model
- End-to-end security
- Unified metadata management

Oracle SOA Suite enables you to transform complex application integrations into agile and reusable service-based applications to shorten the time to market, respond faster to business requirements, and lower costs. Critical business services, such as customer, financial, ordering information, and others that were previously accessible only in packaged application user interfaces can now be rapidly modeled for mobile devices such as smart phones and tablets.

Figure 1-1 provides an overview of Oracle SOA Suite architecture.

Figure 1-1 Oracle SOA Suite Architecture



Key Concepts

Oracle SOA Suite's hot-pluggable architecture enables businesses to reduce costs through reuse of existing IT investments and assets, regardless of the operating system on which they run or the technology on which they are built. Oracle SOA Suite provides easy-to-use, reusable, and unified application development tooling and life cycle management support to further reduce development and maintenance costs and complexity. Businesses can improve efficiency and agility through rules-driven, business process automation with Oracle SOA Suite. Oracle SOA Suite's ability to deliver real-time trending and analysis, visualization, and life cycle visibility enables businesses to anticipate and respond to change when it matters. Oracle SOA Suite provides the following capabilities:

- Unifies cloud applications with on-premises applications to minimize complexity.
- Leverages existing functionality for rapid mobile enabling with representational state transfer (REST) support.
- Designs SOA composite applications from disparate services and applications.
- Connects to virtually any data source technology (messaging, database, and so on), application, or trading partner through a unified connectivity framework, including adapters and B2B gateways, and preintegration with Oracle Data Integration Suite.
- Routes, transforms, and virtualizes services through the highly scalable Oracle Service Bus.
- Orchestrates and builds process automation with Oracle BPEL Process Manager.
- Builds agility by externalizing specific blocks of logic using Oracle Business Rules.
- Gains real-time visibility into operation and performance of business processes, including the ability to respond to specific situations, through Oracle Business Activity Monitoring.
- Consistently and easily secures all services through a policy-driven integrated security framework and the global policy manager in Oracle Enterprise Manager Fusion Middleware Control.

- Executes SOA composite applications through a unified, optimized infrastructure. The SOA service infrastructure is built on top of Oracle WebLogic Server, JRockit, and Oracle Coherence.
- Manages and monitors the previously-mentioned components through a single console natively integrated with Oracle Enterprise Manager Fusion Middleware Control.

Key Components

Oracle SOA Suite includes the following key components.

- Oracle Service Bus

  Oracle Service Bus is a configuration-based, policy-driven enterprise service bus designed for SOA life cycle management. Oracle Service Bus provides the following capabilities:

  – Service discovery and intermediation
  – Rapid service provisioning and deployment
  – Highly-scalable and reliable service-oriented integration, service management, and traditional message brokering across heterogeneous IT environments
  – Intelligent message brokering with routing and transformation of messages, along with service monitoring and administration in a unified software product

  For more information, see *Developing Services with Oracle Service Bus* and *Administering Oracle Service Bus*.

- Oracle Business Process Execution Language (BPEL) Process Manager

  Oracle BPEL Process Manager provides a comprehensive, standards-based, and easy-to-use solution for assembling a set of discrete services into an end-to-end process flow to reduce the cost and complexity of process integration. The BPEL process service engine is a mature, scalable, and robust BPEL server. It executes standard BPEL processes and provides dehydration capability. This enables the state of long-running business flow instances to be automatically maintained in a database, enabling clustering for both failover and scalability. Built-in human workflow services such as task, notification, and worklist management are provided to enable the integration of people and manual tasks into BPEL business flow instances. Oracle BPEL Process Manager can integrate applications and legacy systems, composing coarse-grained services from finer-grained services, building process-centric composite applications, and automating business processes and workflow applications, including routing and escalation.

  For more information, see *Developing SOA Applications with Oracle SOA Suite* and *Administering Oracle SOA Suite and Oracle Business Process Management Suite*.

- Oracle Business Activity Monitoring (BAM)

  Oracle BAM monitors business processes in real time to enable you to make informed tactical and strategic business decisions. Unlike traditional reporting systems, Oracle BAM offers *right-time* operational intelligence for mission critical business processes. Oracle BAM analyzes data before, during, and after business events.

  For more information, see *Monitoring Business Activity with Oracle BAM*.

- Oracle Business Rules

  Oracle Business Rules enable dynamic business decisions at runtime, enabling you to automate policies, computations, and reasoning while separating rule logic from underlying application code. This provides for agile rule maintenance and enables business analysts to modify rule logic without programmer assistance and without interrupting business processes.

For more information, see *Designing Business Rules with Oracle Business Process Management*.

- Oracle Java EE Connector Architecture (JCA) adapters

  Oracle JCA adapters enable connectivity to virtually any data source inside the enterprise. Oracle JCA adapters are standards-based and support both web services and JCA technologies. Oracle JCA Adapters are available for the following:

  – Packaged applications
  – Legacy and mainframe applications, including Tuxedo, Virtual Storage Access Method (VSAM), and Customer Information Control System (CICS)
  – Cloud applications
  – Technologies and protocols, including FTP, files, databases, AQ, JMS, MQSeries, Coherence, LDAP, User Messaging Service, and Oracle E-Business Suite

  For more information, see *Understanding Technology Adapters*.

- Oracle B2B

  Oracle B2B enables an enterprise to exchange information electronically with a trading partners. Oracle B2B supports a set of industry standards, including Electronic Data Interchange (EDI), UCCnet, RosettaNet, Chemical Industry Data Exchange (CIDX), Petroleum Industry Data Exchange (PIDX), Voluntary Interindustry Commerce Solutions (VICS), ebXML, and Universal Business Language (UBL).

  For more information, see *Using Oracle B2B*.

- Oracle SOA for Healthcare

  Oracle SOA Suite for Healthcare enables you to design, create, and manage applications that process health care data. Oracle SOA Suite for Healthcare integration provides a web-based user interface in which to create and configure health care integration applications, and monitor and manage the messages processed through those applications. You can also use the Oracle Document Editor to create and configure document definitions that define message structures.

  For more information, see *Using Oracle SOA Suite for Healthcare Integration*.

- Oracle Web Services Manager (OWSM)

  OWSM provides the policy manager for securing web services, including authentication and authorization. OWSM is installed by default when you install Oracle Fusion Middleware Infrastructure. It is licensed only through Oracle SOA Suite; a standalone license is not available.

  For more information, see Enabling Security with Policies and Message Encryption in *Developing SOA Applications with Oracle SOA Suite*.

Oracle SOA Suite can be integrated with the following additional components:

- Oracle Enterprise Scheduler

  Enterprise applications require the ability to off-load large transactions to run at a future time or automate the running of application maintenance work based on a defined schedule. Oracle Enterprise Scheduler enables you to run different job types, including Java, PL/SQL, binary scripts, web services, and Enterprise JavaBeans (EJBs) distributed across the nodes in an Oracle WebLogic Server cluster. Oracle Enterprise Scheduler runs jobs securely, with high availability, scalability, and load balancing. Oracle Enterprise Scheduler runs are monitored and managed through Oracle Enterprise Manager Fusion Middleware Control.

For more information, see *Developing Applications for Oracle Enterprise Scheduler* and *Administering Oracle Enterprise Scheduler*.

- Oracle Managed File Transfer

  Oracle Managed File Transfer is a high performance, standards-based, end-to-end managed file gateway. It features design, deployment, and monitoring of file transfers using a lightweight, web-based, design-time console that includes file encryption, scheduling, and embedded FTP and sFTP servers.

  For more information, see *Using Oracle Managed File Transfer*.

Key Management Tools and Processes

Oracle SOA Suite provides a number of development, monitoring, and management tools.

Oracle JDeveloper

Oracle JDeveloper is the integrated development environment used by Oracle SOA Suite for building service-oriented applications with the latest industry standards for Java, XML, web services, SQL, REST, and SCA. Oracle JDeveloper supports the complete development life cycle with integrated features for modeling, coding, debugging, testing, profiling, tuning, and deploying applications. Oracle JDeveloper features a SOA Composite Editor for quickly and graphically assembling the various components and technologies used in a SOA project. User friendly wizards are provided to simplify many common tasks such as connecting to IT systems.

Oracle Enterprise Manager Fusion Middleware Control

Oracle Enterprise Manager Fusion Middleware Control is a web-based tool for managing and monitoring SOA composite applications at runtime. Administrators perform tasks such as tracking business flow instances, attaching security policies, identifying a specific message by searching on specific data, identifying and repairing errors in the Error Hospital, and so on. Oracle Enterprise Manager Fusion Middleware Control also provides visibility into the execution of processes, showing a complete end-to-end graphical representation of the business flow followed by a given message across all the components traversed.

Additional Design and Runtime Tools

Oracle SOA Suite provides additional design and runtime tools for some components, as described in Table 1-1.

Table 1-1 Additional Design and Runtime Tools

Runtime Tools	Description
Oracle BAM Composer	Provides a user interface for creating dashboards, alerts, business views, key performance indicators (KPIs), alerts, and parameters.
Oracle B2B Console	Provides a user interface for creating B2B transactions, including trading partners and trading partner agreements.
Oracle Managed File Transfer Console	Provides a lightweight, web-based, design-time console for defining the following artifacts: • The origin of files to transfer (known as sources) • The destination of files (known as targets) • A transfer that associates a source with targets

Table 1-1 (Cont.) Additional Design and Runtime Tools

Runtime Tools	Description
Oracle Service Bus Console	Provides configuration tools for creating service level agreement alerts, pipeline alerts, messaging reporting actions, alert destinations, and throttling groups for business service endpoints. You can also update environmental values, either individually or in bulk.
Oracle SOA Composer	Provides a runtime environment for creating domain value maps, approval management extensions, business rules, and composite sensors in deployed composites.
Oracle Healthcare for Healthcare Integration Console	Provides support for messaging protocols and creating and managing endpoints, managing documents, creating map sets, and creating Java callouts.

Overview of an Oracle SOA Suite Process Flow

The remaining chapters of this guide provide an overview of how Oracle SOA Suite components work together in a process flow from design time through runtime to address the business challenges faced by a company.

For more information, see Business Challenges of Company X .

Part II
Business Challenges and Oracle SOA Suite

This part describes a business challenge and how the components of Oracle SOA Suite address these challenges from design time through runtime.

2

Business Challenges of Company X

This chapter describes the business challenges faced by Company X and how Oracle SOA Suite provides a business solution for these challenges.
This chapter includes the following sections:

Business Challenges of Company X

Company X must improve their order processing system to accommodate multichannel growth with online business partners. In addition, an aggressive store expansion is planned. Overlapping systems must be consolidated to provide better end-to-end visibility from order to fulfillment.

Requirements for improving the order processing system are as follows:

- The order processing system must be accessible through multiple protocols, data formats, and client types, including mobile devices:
  - Business trends indicate that Company X must launch a mobile application soon and the new order processing service must support access through RESTful APIs.
  - In addition to the existing online direct store, Company X plans to launch a service in which orders are received through a different channel (as batch comma-separated value (CSV) files over FTP). They must eventually be processed and fulfilled using the same new order provisioning infrastructure.
  - Company X must interface with trading partners and provide electronic data interface (EDI) support.
- For large orders, a customer's credit history must be checked before sending the order for fulfillment. Otherwise, the order is rejected. Initially, credit is checked by internal departments, but later must be integrated with PayPal. Changing credit providers must not disrupt order processing operations.
- The order processing system must provide direct integration with the packaging department to ship orders with preferred shipping providers based on the type of shipping service (2 day, 5-7 day shipping, and so on).
- The bulk fulfillment process must run according to a predefined pick-up schedule.
- Upon fulfillment processing and orders being sent to the packaging department, a message must be communicated to the customer (either bulk or on-demand).

Solutions

The chapters of this guide describe how Company X uses the capabilities of Oracle SOA Suite to address their business challenges. Table 2-1 provides an overview.

Table 2-1 Addressing Business Challenges

Challenge Addressed	See...
Company X designs a credit card validation composite to validate payments and return payment status. If a payment is denied, an order is not processed. Oracle Service Bus is integrated with the composite to provide registration and security benefits.	Creating a Credit Validation System
Company X designs a SOA composite application to accept new purchase orders, authorize or deny them, and forward the authorized orders to an order fulfillment composite designed in Fulfilling Orders . Oracle Service Bus makes the order processing composite available over many protocols and data formats, and validates the order.	Creating an Order Processing System
Company X designs an Oracle Service Bus pipeline to connect a proxy service to a file ordering channel. The proxy handles incoming customer orders by file.	Adding New Ordering Channels with Oracle Service Bus
Company X designs a BPEL process that sets the status of an order to shipped, notifies the customer that the order has shipped, and updates the order status in the database. This process is connected to an inbound REST interface service that defines a shipping resource.	Packing and Shipping Orders
Company X designs an order fulfillment composite to listen for orders to process, selects a shipping provider, and invokes the packing and shipping service designed in Packing and Shipping Orders .	Fulfilling Orders
Company X designs a query inventory composite to identify the total number of items for each product ordered daily for a given category. Oracle Enterprise Scheduler is used to define a web service job for the query inventory composite and then submit the job with a schedule to run at a specified time.	Scheduling Composite Execution
Company X designs an Oracle Managed File Transfer flow to receive files and write them to a file system using the Managed File Transfer embedded FTP server. Oracle Managed File Transfer invokes a Managed File Transfer service in a composite and dynamically decides based on file size whether to pass the content inline or by reference.	Managing File Transfers
Company X designs BPEL process analytic measurements and business indicators on activities in a BPEL process. Oracle BAM Composer is used to create a dashboard of these analytics to gain business insights into customer order requests.	Gaining Business Insights with Oracle Business Activity Monitoring

3

Creating a Credit Validation System

This chapter describes how Oracle SOA Suite addresses the business challenge of creating a credit validation system. Overviews of how key SOA composite application components are created and address this challenge are provided, including BPEL process invoke, assign, and transformation activities; database adapters; SOA templates; and composite sensors. The role of Oracle Service Bus in this business solution is also described.
This chapter includes the following sections:

Business Challenge

Company X has embarked upon a project to improve customer satisfaction. A key area for improvement is the need to streamline the ordering process to provide better order tracking visibility through the following parts of the ordering life cycle:

- Credit approvals
- Fulfillment
- Shipment
- Delivery

A key issue in their current system is that credit card payments are often denied for minor reasons. Since the process to correct these issues varies across Company X's order entry systems, there is no consistent follow-up and resolution to customers. Orders become lost and delayed in the system, causing customer dissatisfaction.

Company X has determined that a new credit card fraud detection system must also be in place at year's end to eliminate credit card abuses. A consistent fraud mechanism requires the credit validation process to be consolidated across all order entry systems.

The first step is to provide a consistent interface for all order entry applications for credit validation. The consolidated credit validation service is to be initially hosted in-house to control quality. However, once the interface is stabilized, this service is to be outsourced to a third party provider. In the future, when Company X decides to outsource credit validation to an external provider, this can be accomplished without impacting existing applications.

Business Solution

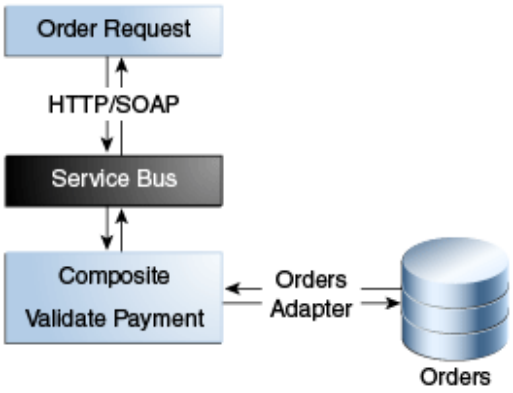
To address this business challenge, Company X designs a business solution that uses the components described in Table 3-1.

Table 3-1 Components That Provide the Business Solution

Component	How This Component Addresses The Business Challenge	Component Description
SOA composite application	A SOA composite application is designed to validate payments and return status. If payment is denied, an order is not processed. The composite consists of the following components, each of which is briefly described below: • BPEL process service component • Database adapter • SOA template • Composite sensor	SOA composite applications consist of the following: • Service binding components: Provide the outside world with an entry point to the SOA composite application. • Service components Implement the business logic or processing rules of the application. • Reference binding components: Enable messages to be sent from the SOA composite application to external services in the outside world.
• BPEL process service component	The BPEL process (and its activities) orchestrates the validation of credit card payments and returns a payment status (invokes a database adapter).	BPEL processes provide a comprehensive and easy-to-use infrastructure for creating, deploying, and managing business processes. BPEL is the standard for assembling discrete services into an end-to-end process flow. BPEL processes orchestrate and build process automation.
• Database adapter	The database adapter stores and retrieves credit card payment information from the database, including payment type, card number, expiration date, card name, and daily limit. If the credit card number is not available in the database, payment is denied.	The database adapter enables Oracle SOA Suite and Oracle Fusion Middleware to communicate with database endpoints. These include Oracle database servers and any relational databases that follow the ANSI SQL standard and provide JDBC drivers.
• SOA template	A BPEL scope activity template is imported that includes a transformation activity that determines the payment status (authorized or denied) based on the daily limit and total order amount.	A SOA template is a reusable part of an Oracle SOA Suite project that you use to create new projects. There are three types of templates: • SOA project • Service component • Custom BPEL scope activity
• Composite sensor	The composite sensor tracks credit card payment status.	Composite sensors provide a method for implementing trackable fields on messages.
Oracle Service Bus	Oracle Service Bus provides the following composite registration and security benefits: • A business service registers the composite URI of the SOA composite application. • A pipeline validates the SOA composite application before invocation. • A proxy enables customers to invoke the composite through a proxy instead of connecting directly to the composite.	Oracle Service Bus is a configuration-based, policy-driven enterprise service bus. It provides highly scalable and reliable service-oriented integration, service management, and traditional message brokering across heterogeneous IT environments.

Figure 3-1 provides an overview of how this business solution is implemented.

Figure 3-1 Payment Validation Overview



Subsequent sections of this chapter provide more specific details about how the components in Table 3-1 are used to address the credit validation business challenge.

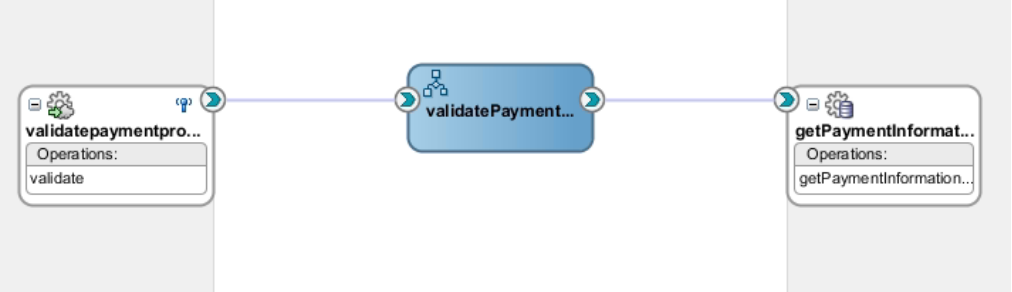
- Creating a Credit Validation Composite
- Retrieving Credit Card Payment Information from the Database
- Invoking the Database Adapter from the BPEL Process
- Calculating Payment Status with XSLT Transformations
- Tracking Payment Status with Composite Sensors
- Deploying the validatePayment Composite
- Registering SOA Composite Applications with Oracle Service Bus
- Deploying and Testing

Creating a Credit Validation Composite

The business solution is designed in a SOA composite application that validates a credit card payment and returns a payment status. If the payment is denied, the order is not processed. The implementation of this service uses a BPEL process to invoke a database adapter to retrieve the credit card data from the database and perform the validation. The service returns a payment status of either authorized or denied.

Figure 3-2 provides an overview of the credit validation composite. Customer requests come through the **validatepaymentprocess** service. The SOA composite application (named **validatePayment**) takes these requests and invokes a database adapter reference named **getPaymentInformation** to retrieve credit card information from the database. The database adapter is configured with the Adapter Configuration Wizard.

Figure 3-2 Credit Validation Process



The following example shows the content of an inbound order message. The customer provides their credit card number, expiration date, card type, and billing address.

```
<soas:Billing >
   <soas:CardPaymentType>1</soas:CardPaymentType>
   <soas:CardNum>1234123412341234</soas:CardNum>
   <soas:ExpireDate>0316</soas:ExpireDate>
   <soas:CardName>AMEX</soas:CardName>
   <soas:BillingAddress>
          <soas:FirstName>Joe</soas:FirstName>
          <soas:LastName>Smith</soas:LastName>
          <soas:AddressLine>555 Beverly Lane</soas:AddressLine>
          <soas:City>Hollywood</soas:City>
          <soas:State>CA</soas:State>
          <soas:ZipCode>12345</soas:ZipCode>
          <soas:PhoneNumber>5127691108</soas:PhoneNumber>
   </soas:BillingAddress>
</soas:Billing>
```

Retrieving Credit Card Payment Information from the Database

All available credit card details are stored in a database, including payment type, card number, expiry date, card name, and daily limit. The database adapter retrieves credit card payment information from the database, using the customer's credit card number as the key.

The validation process includes three steps:

- The payment information is first retrieved from the database, using the credit card number quoted in the order message as the key. If there is no data available with this credit card number, payment is denied.
- If data for the credit card number is available, the expiration date in the database record is compared to the expiration date in the order message. If they are not the same, the payment is also denied.
- The last check compares if the total order amount is less than the daily limit on the credit card in the database.

If all tests are successful, payment is authorized. Otherwise, payment is denied.

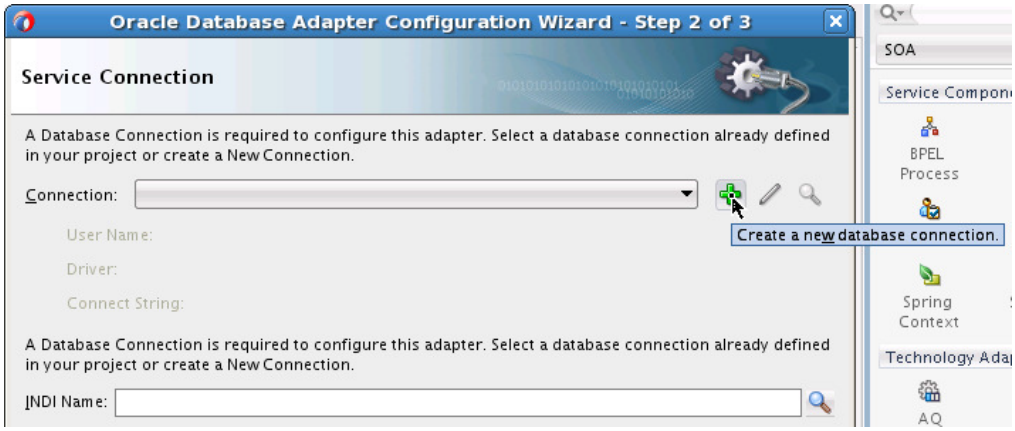
To invoke the Adapter Configuration wizard, the database adapter is dragged from the Components window into the **External References** swimlane of the SOA Composite Editor. This action invokes the Adapter Configuration Wizard for configuring the database adapter.

Configuring the Database Adapter with the Adapter Configuration Wizard

The following database adapter configuration tasks are performed during execution of the Adapter Configuration Wizard.

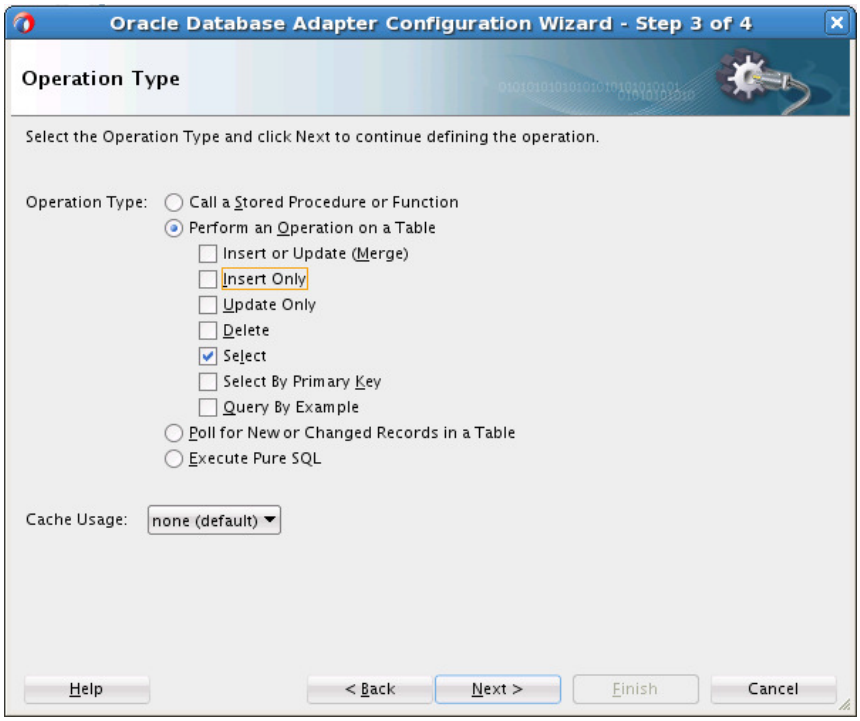
- The option to create a connection to the database in which the credit card information is stored is selected, as shown in Figure 3-3. This enables the database adapter to access the database and retrieve the appropriate credit card information.

Figure 3-3 Database Connection



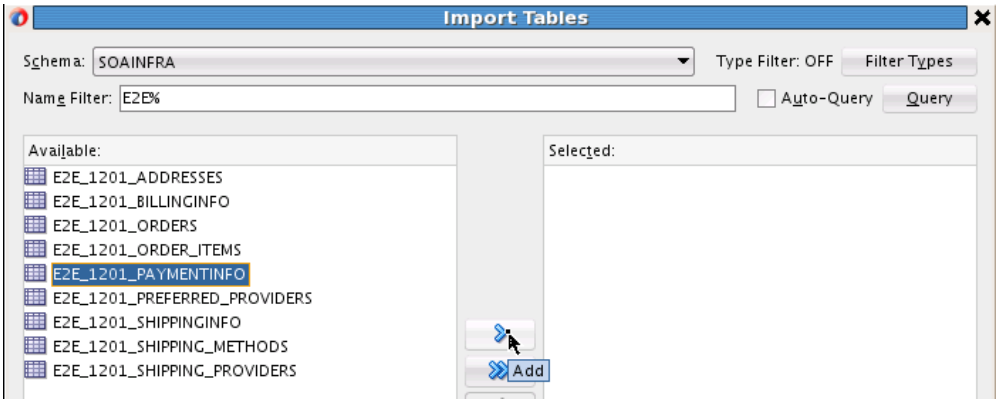
- The **Select** database option is chosen to create a database query, as shown in Figure 3-4.

Figure 3-4 Database Operation Type



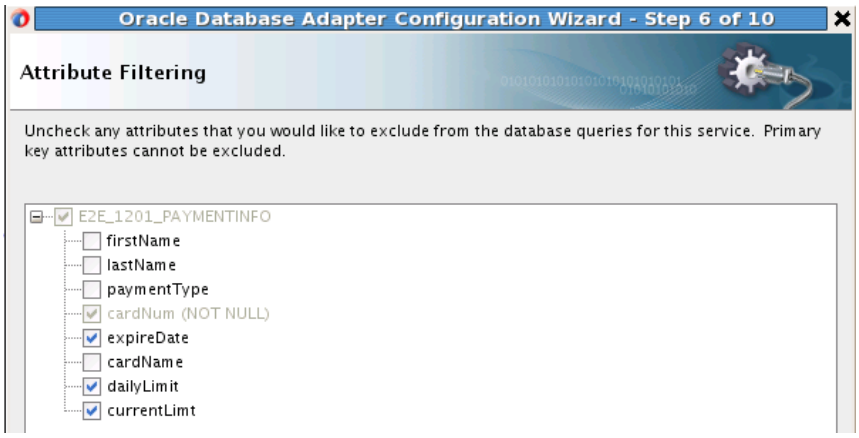
- The appropriate credit card information table is imported from the database, as shown in Figure 3-5.

Figure 3-5 Table with Credit Card Information Imported



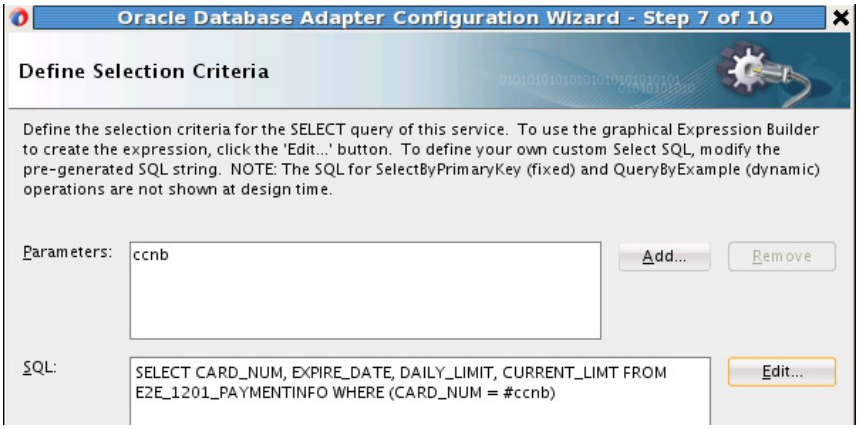
- The appropriate credit card filtering information to use in creating the **Select** operation database query is enabled, as shown in Figure 3-6.
  - Expiration date
  - Daily limit
  - Current limit

Figure 3-6 Credit Card Filtering



- The appropriate credit card criteria for the **Select** operation is specified, as shown in Figure 3-7. The **ccnb** parameter is provided for the customer's credit card number that is specified in the inbound message (`CardNum`), as shown in Creating a Credit Validation Composite. The database adapter retrieves credit card payment information from the database, using the credit card number as the key.

Figure 3-7 Define Selection Criteria

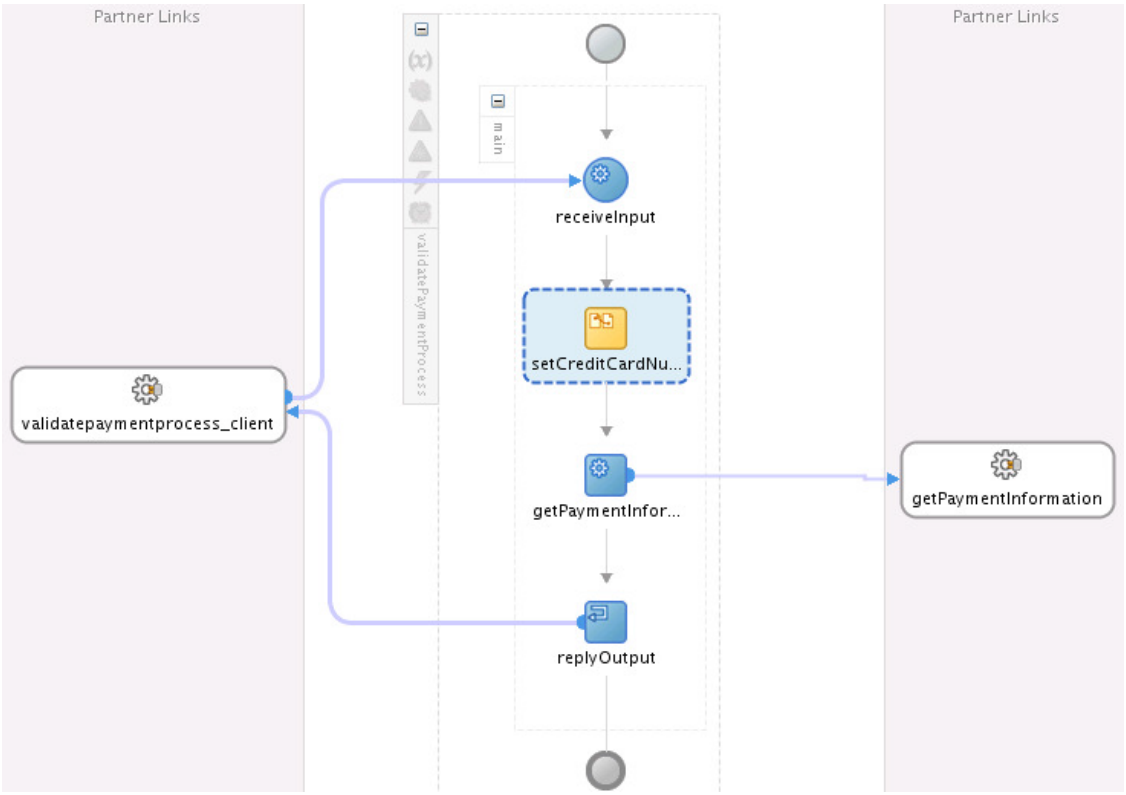


The database adapter processes the selections and generates a reference that implements the operation specified. The SOA composite application now contains the WSDL file to represent the database adapter reference: **getPaymentInformation.wsdl**. The composite diagram shown in Figure 3-2 shows the **getPaymentInformation** reference.

Invoking the Database Adapter from the BPEL Process

Company X creates a BPEL process service component in the SOA composite application. The BPEL process orchestrates the logic of the business solution. An invoke activity in the BPEL process invokes the database adapter as a partner link to retrieve credit card information from the database. Figure 3-8 provides details.

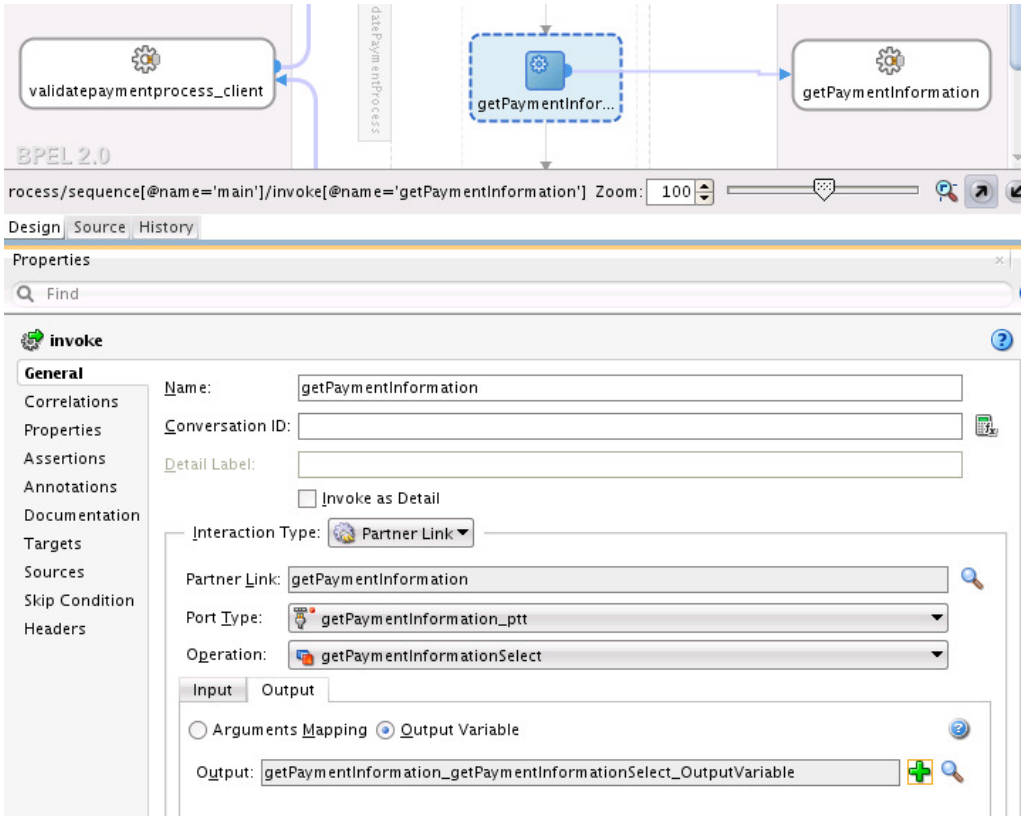
Figure 3-8 BPEL Process Invocation of the getPaymentInformation Reference



Within the BPEL process, the invoke activity calls the database adapter (named **getPaymentInformation**), as shown in Figure 3-9. You can create and edit invoke activities details from the Property Inspector below the designer or by double-clicking the invoke activity.

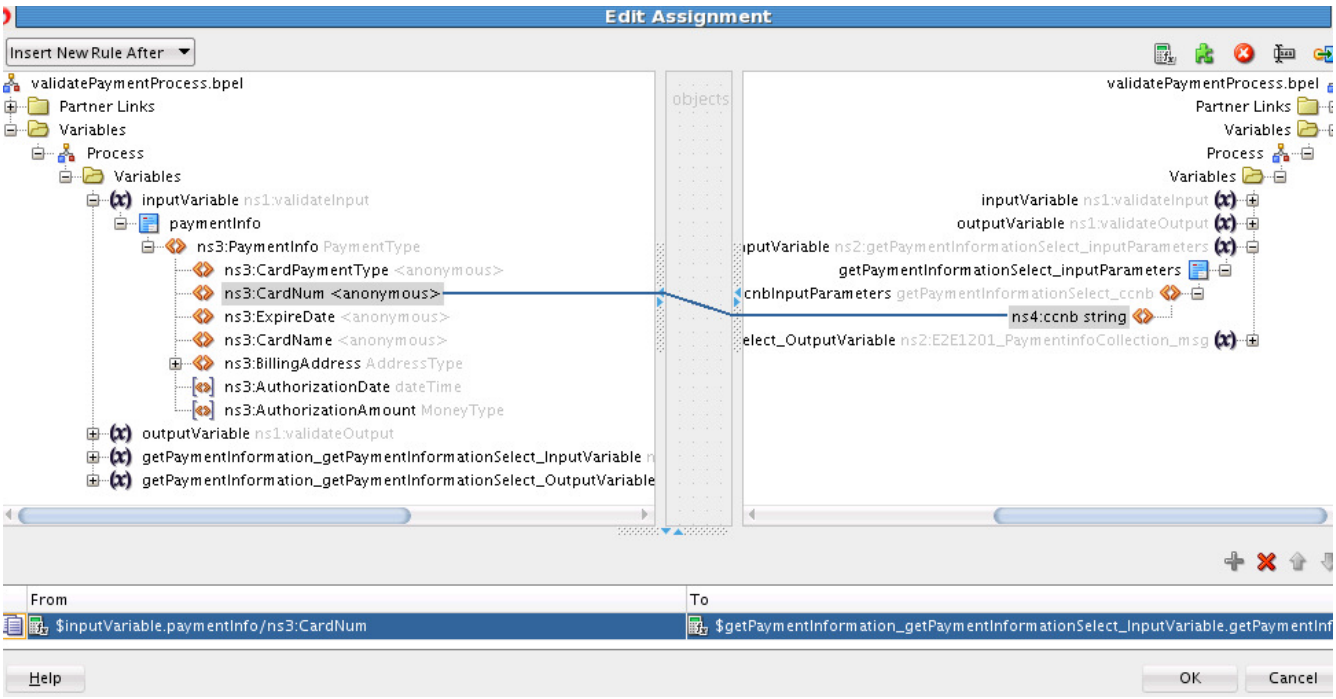
- Name
- Partner link to invoke (for this example, the database adapter)
- Port type
- Operation to perform
- Input variable under the **Input** tab (contains the credit card number to send to the database adapter)
- Output variable under the **Output** tab (returns the results from the database adapter)

Figure 3-9 Invoke Activity Invoking the Database Adapter



An assign activity is used to populate the input variable in the invoke activity. In the **Copy Rules** tab of the assign activity (named **setCreditCardNumber**), the credit card number passed into the BPEL process as **CardNum** is assigned to the **ccnb** parameter of the **getPaymentInformation** database adapter. Figure 3-10 provides details.

Figure 3-10 Credit Card Number Assigned to the Input Variable



An XSLT map determines if the payment is valid based on the information returned by the database adapter.

Calculating Payment Status with XSLT Transformations

Company X requires a mechanism for determining if payment status is authorized or denied. To address this requirement, an XSLT transform activity determines if the payment is valid based on the following information returned by the database adapter.

- The daily limit (retrieved from the database)
- The total order amount (authorization amount in the order message, which has been calculated in the process order project by multiplying the price and amount of every order item and adding them). The total amount of the order must be smaller than the daily limit on the credit card.

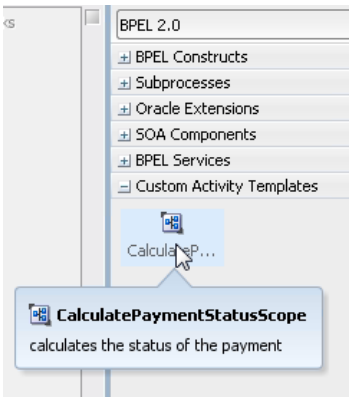
XSLT transformation design can be packaged in the following ways:

- Within an individual transform activity in a BPEL process
- As part of a template. Templates enable you to share common code between applications, composites, and processes. You create a template once, then share it as needed. The template can be reused multiple times. Three types of templates are supported:
  - Project templates that provide a complete project with all components and resources.
  - Service component templates such as a BPEL process with all references and components.
  - Custom activity templates that consist of a BPEL process scope activity.

Company X decides to use a template. For this example, a custom activity template is created and then imported that consists of a scope activity with a transform activity.

You create and design templates in Oracle JDeveloper. After creation, templates are displayed in the Components window for selection and use, as shown in Figure 3-11. The custom activity template can be dragged into a BPEL process, as needed.

Figure 3-11 Custom Activity Template

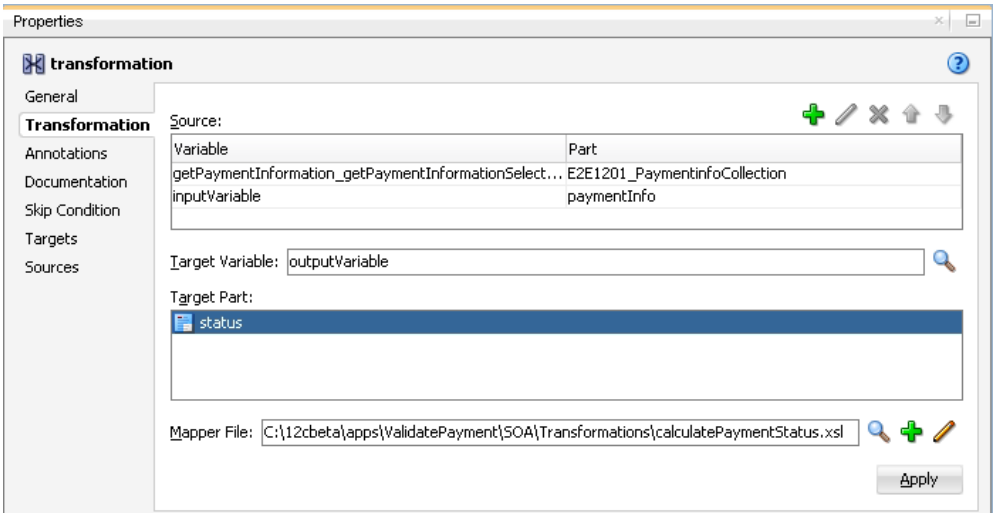


The custom scope activity consists of the transform activity. The transformation for this example expects two input variables, as shown in the **Sources** section of Figure 3-12.

- An output variable of the database adapter, which includes the payment information stored in the database.
- An input variable of the BPEL process, which includes the total order amount.

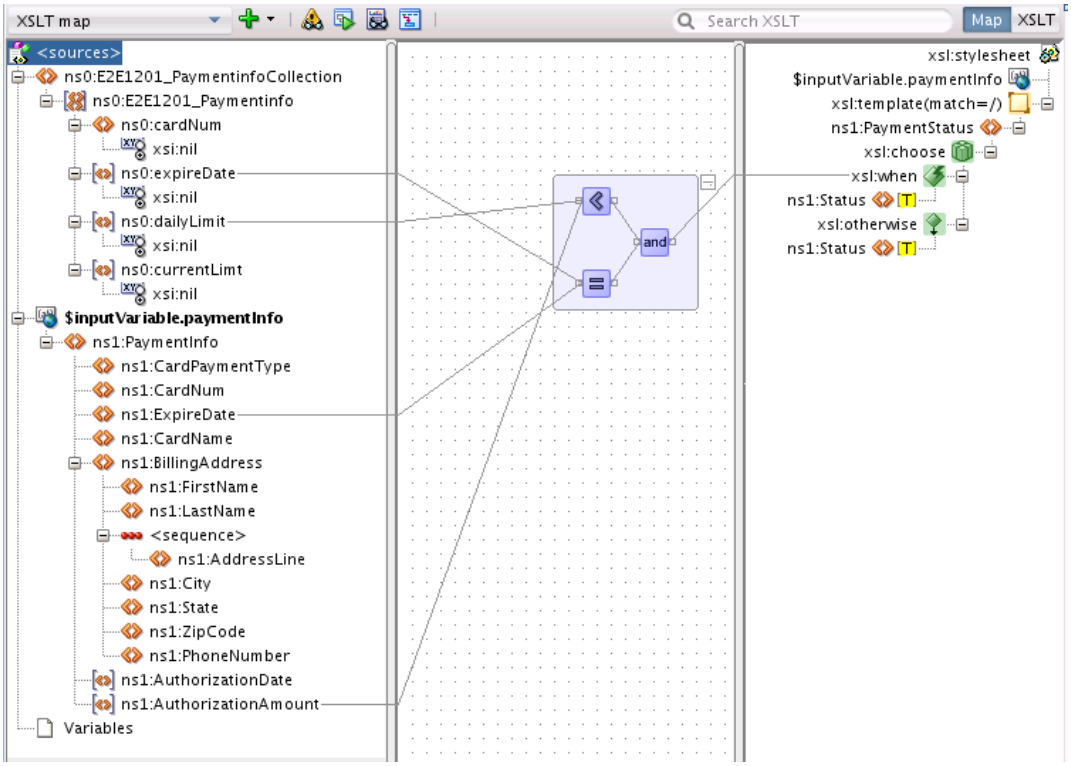
The output is in the **status** field of the output message, as shown in the **Target Part** section of Figure 3-12. This field is either set to **Denied** or **Authorized**.

Figure 3-12 Transformation Variables



The mapping in the XSLT Map Editor is shown in Figure 3-13.

Figure 3-13 Transformation in the XSLT Map Editor



Tracking Payment Status with Composite Sensors

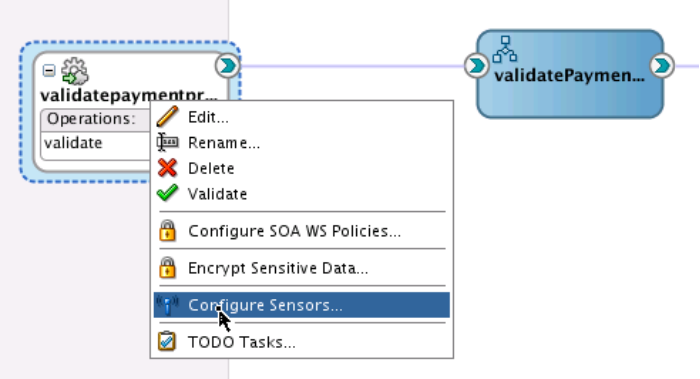
Company X must be able to track the status of order payments (either authorized or denied). To address this requirement, composite sensors are used. Composite sensors provide a method for implementing trackable fields on messages. Composite sensor data is persisted in the database during runtime, enabling you to search for all authorized or denied payments.

Composite sensors enable you to perform the following tasks:

- Monitor incoming and outgoing messages.
- Locate particular instances by searching for specific sensor details in Oracle Enterprise Manager Fusion Middleware Control.
- Publish JMS data computed from incoming and outgoing messages.
- Track composite instances initiated through business event subscriptions.

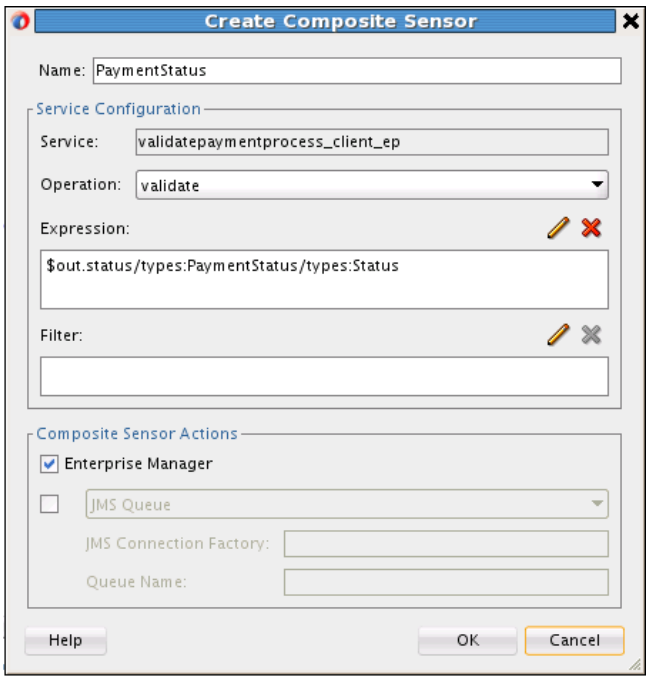
The composite sensor is defined on the inbound SOAP web service binding component, as shown in Figure 3-14. You can also define composite sensors on reference binding components and service components that have business event subscriptions.

Figure 3-14 Composite Sensor Definition on a SOAP Service Binding Component



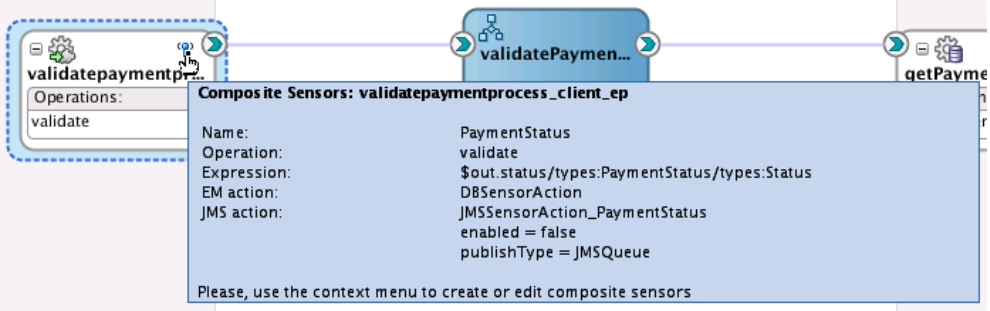
The Create Composite Sensor dialog shown in Figure 3-15 shows that an XPath expression is defined to track the payment status (authorized or denied). The **Enterprise Manager** check box is also selected. This enables you to view composite sensor names and values (for example, **Status=Authorized**) in the Flow Instances page of Oracle Enterprise Manager Fusion Middleware Control.

Figure 3-15 Composite Sensors



The sensor is displayed as an icon on the **validatepaymentprocess** SOAP web service binding component. When you place your cursor over the icon, the composite sensor definition is displayed. Figure 3-16 provides details.

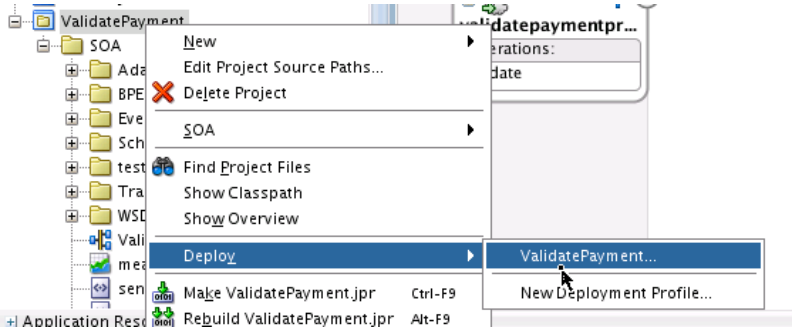
Figure 3-16 Composite Sensor Details



Deploying the validatePayment Composite

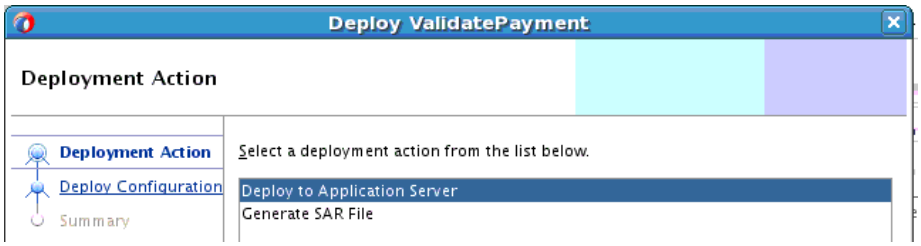
Company X deploys the **validatePayment** composite in Oracle JDeveloper, as shown in Figure 3-17.

Figure 3-17 validatePayment Composite Deployment



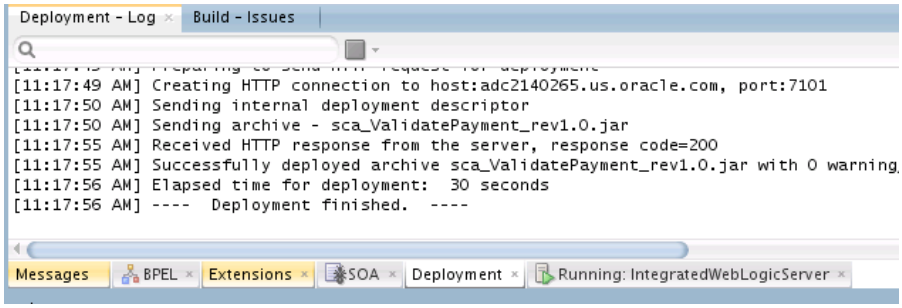
During the deployment process, the composite is deployed to the server on which an application server connection was created in Configuring the Database Adapter with the Adapter Configuration Wizard. Figure 3-18 provides details.

Figure 3-18 Deployment to the Application Server



If there are no compilation errors, the build is successful and deployment starts. Figure 3-19 provides details.

Figure 3-19 Deployment Success Message in Oracle JDeveloper Log Window

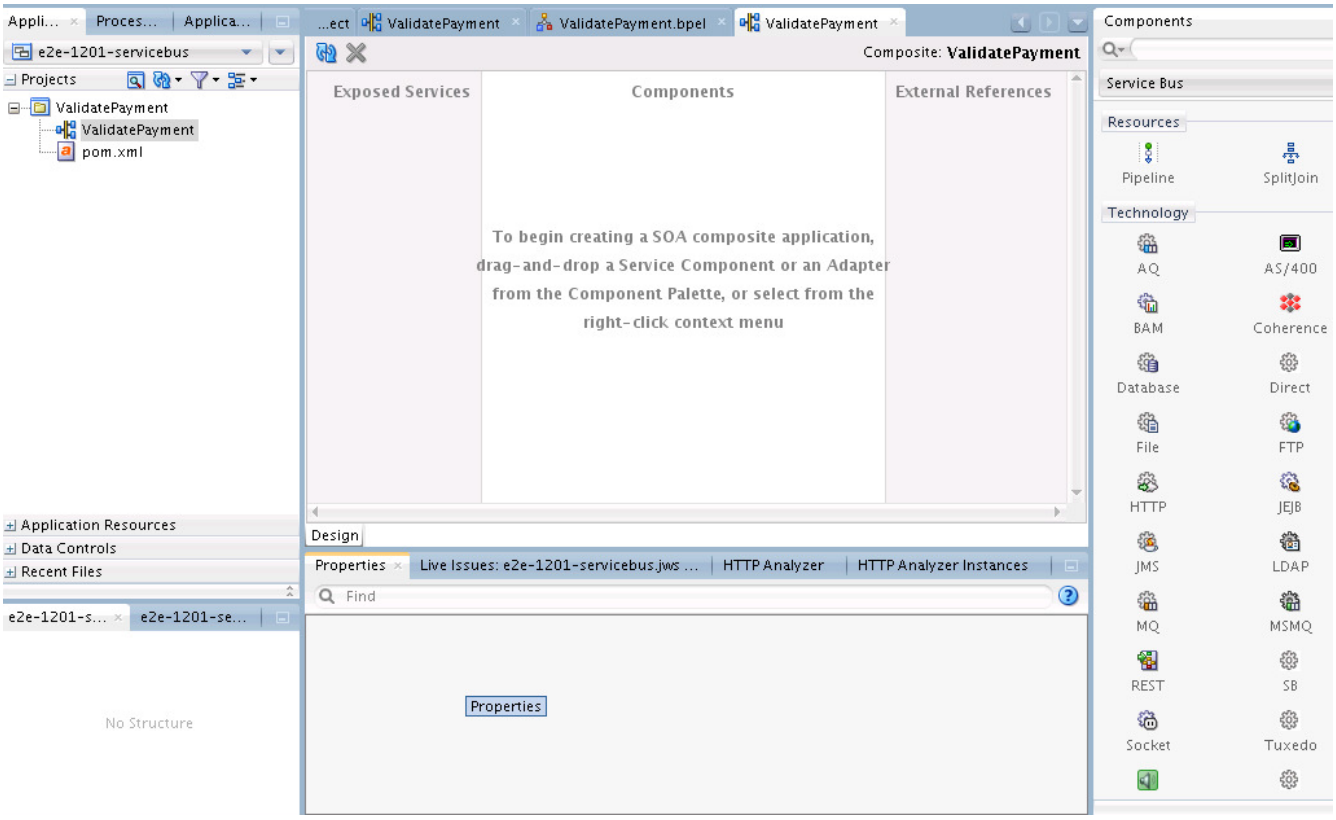


Registering SOA Composite Applications with Oracle Service Bus

Company X must be able to protect **validatePayment** composite users from routine changes such as deployment location and implementation updates. To address this requirement, Company X registers the **validatePayment** composite with Oracle Service Bus. Oracle Service Bus scales the service to handle higher volumes of requests and provides resiliency for the service if it is taken down for routine maintenance.

Company X begins by creating an Oracle Service Bus application in which to perform registration. Figure 3-20 provides details. Company X can create proxies, pipelines, and business services by dragging icons from the Components window on the right into the designer. In the Components window, pipeline and split-join icons are displayed. In Release 12*c*, the pipeline is separated from the proxy to enable it to be a reusable component. Adapters and transports are also displayed for building business services (in the **External References** swimlane) and proxies (in the **Exposed Services** swimlane).

Figure 3-20 Oracle Service Bus Application Ready for Design Registration



When design registration is complete, the business solution looks as shown in Figure 3-21.

Figure 3-21 Oracle Service Bus Application Registration

The following sections provide an overview of Oracle Service Bus application registration:

- Sharing Resources with Folders
- Registering the Composite URI of the SOA Composite Application
- Configuring Pipelines and Proxies

Sharing Resources with Folders

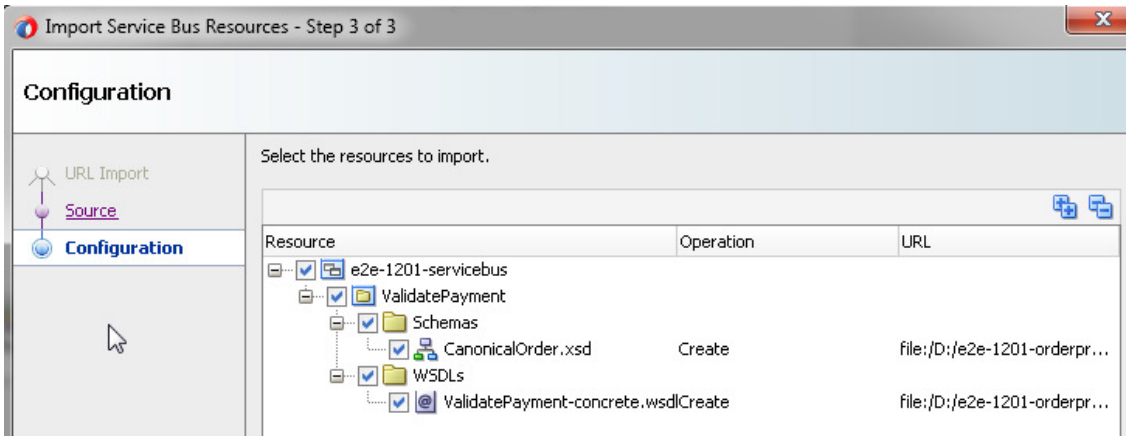
Company X creates folders into which resources such as XSD and WSDL files are imported. This action is performed by right-clicking the Oracle Service Bus application and selecting **New** > **From Gallery** > **Folder** to invoke the Create Folder wizard.

Oracle Service Bus folders categorize resources and are aligned with the default **Schema** and **WSDL** folders that are displayed in the Applications window for the SOA composite

application. Folders provide a way to share resources between Oracle Service Bus and the SOA composite application.

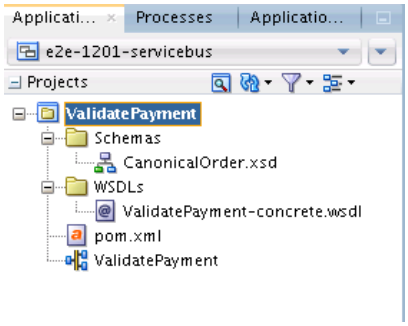
After folder creation, Company X imports the artifacts from the file system for building services by selecting **File** > **Import** > **Service Bus Resources** to invoke the Import Service Bus Resources wizard. When artifact selection is complete, the wizard looks as shown in Figure 3-22.

Figure 3-22 Import Service Bus Resources Wizard - Configuration Page



When wizard configuration is complete, Figure 3-23 shows the artifacts in the Applications window.

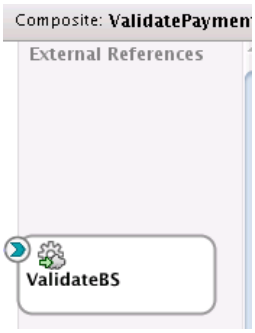
Figure 3-23 Oracle Service Bus Imported WSDL and Schema File Resources



Registering the Composite URI of the SOA Composite Application

A business service (named **ValidateBS** in Figure 3-21) registers the composite URI of the SOA composite application and provides a representation of the **validatePayment** composite. Configuration is performed by dragging an **HTTP** icon from the Components window into the **External References** swimlane of the Oracle Service Bus application, as shown in Figure 3-24.

Figure 3-24 ValidateBS Business Service



This action invokes the Create Business Service wizard for configuring the following:

- HTTP as the transport type
- WSDL as the service type
- The WSDL file
- The endpoint URI is set to the **validatePayment** composite. The endpoint URI format is based on the transport protocol you selected (HTTP). For example:

  ```
  http://localhost:7101/soainfra/services/default/ValidatePayment/
  validatepaymentprocess_client_ep
  ```

Company X double-clicks the **ValidateBS** business service to review the general, transport, performance (result caching), and security policy settings.

Configuring Pipelines and Proxies

A pipeline (named **ValidatePP** in Figure 3-21) contains actions performed on the service bus such as error handling reporting, data transformation, and validation before invoking the composite.

Users invoke the **validatePayment** composite through a proxy (named **ValidatePS** in Figure 3-21) rather then connecting directly to the composite. This provides for more agility and flexibility in managing changes. The proxy is the interface to the service from external consumers.

Pipeline and proxy configuration is performed by dragging a **Pipeline** icon from the Components window into the **Components** section of the Oracle Service Bus application. This action invokes the Create Pipeline Service wizard for pipeline and proxy configuration. During configuration, Company X selects the WSDL file, as shown in Figure 3-25.

Figure 3-25 WSDL Selection for Proxy

The **Expose as Proxy Service** check box is also selected on subsequent pages of the wizard. Figure 3-26 provides details.

Figure 3-26 Expose as a Proxy Service Check Box

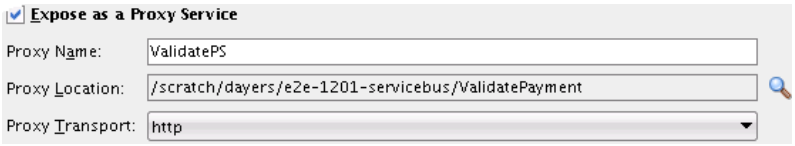
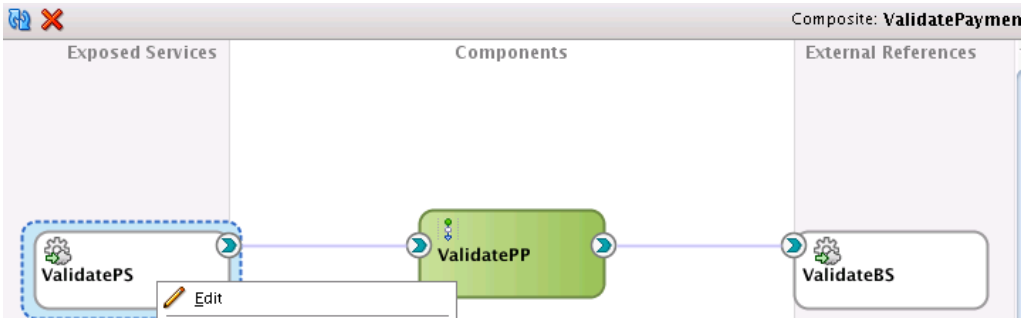


Figure 3-27 shows the **ValidatePP** pipeline in the **Components** section and the **ValidatePS** proxy in the **Exposed Services** swimlane.

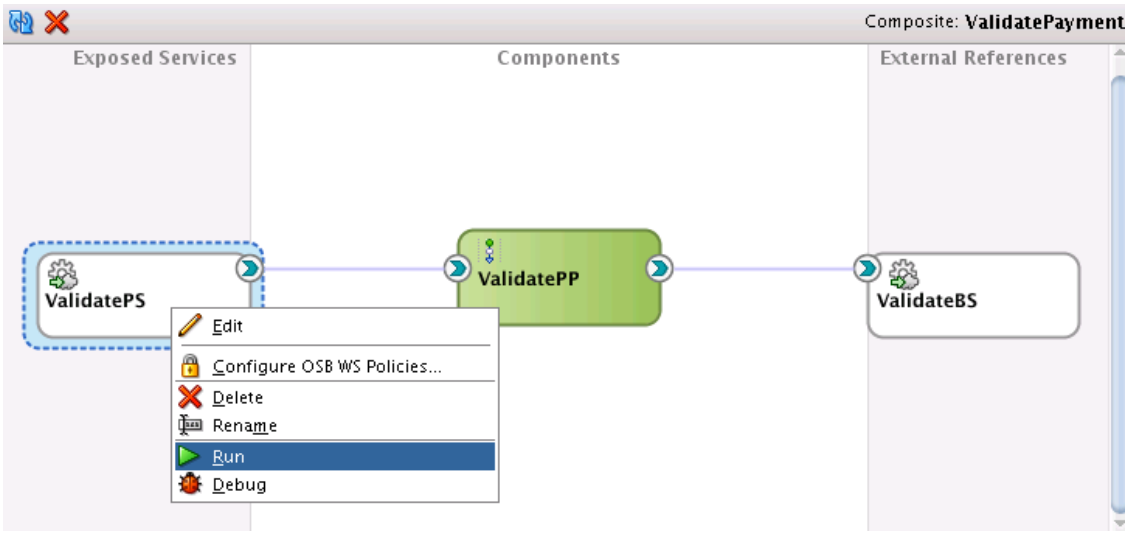
Figure 3-27 Proxy and Pipeline Configuration in the Components Section



Deploying and Testing

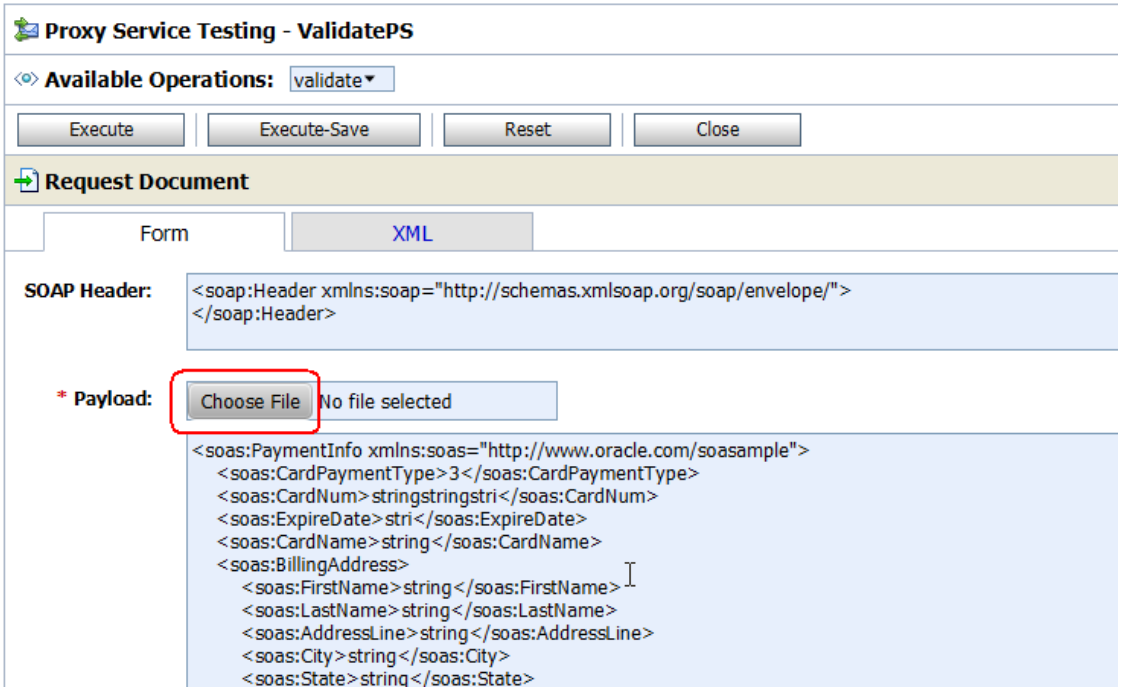
After application design is complete, Company X deploys and tests the application end-to-end by right-clicking the **ValidatePS** proxy and selecting **Run**. Figure 3-28 provides details.

Figure 3-28 Application Invocation



The Test Console is displayed from which you can select a payload for testing. A sample payload is generated for you. Figure 3-29 provides details.

Figure 3-29 Test Console



Company X tests with a sample authorize payment file, as shown in the following example:

```
<ns1:PaymentInfo xmlns:ns1="http://www.oracle.com/soasample">
        <ns1:CardPaymentType>0</ns1:CardPaymentType>
        <ns1:CardNum>1234123412341234</ns1:CardNum>
        <ns1:ExpireDate>0316</ns1:ExpireDate>
        <ns1:CardName>AMEX</ns1:CardName>
        <ns1:BillingAddress>
                <ns1:FirstName>Joe</ns1:FirstName>
                <ns1:LastName>Smith</ns1:LastName>
                <ns1:AddressLine>555 Beverly Lane</ns1:AddressLine>
                <ns1:City>Hollywood</ns1:City>
                <ns1:State>CA</ns1:State>
                <ns1:ZipCode>12345</ns1:ZipCode>
                <ns1:PhoneNumber>5127691108</ns1:PhoneNumber>
        </ns1:BillingAddress>
        <ns1:AuthorizationAmount>100</ns1:AuthorizationAmount>
</ns1:PaymentInfo>
```

After importing the sample file, Company X clicks **Execute**.

Credit validation design and testing is now complete.

Related Documentation

Table 3-2 provides references to documentation that more specifically describes the components and features described in this chapter.

Table 3-2 Related Topics

For Information About...	See...
Creating a SOA composite application	Creating a SOA Application in *Developing SOA Applications with Oracle SOA Suite*
Creating a database connection	Creating an Application Server Connection for Oracle JCA Adapters in *Developing SOA Applications with Oracle SOA Suite*
Configuring a database adapter with the Adapter Configuration wizard	Oracle JCA Adapter for Database in *Understanding Technology Adapters*.
Designing transformations with the XSLT Map Editor	Creating Transformations with the XSLT Map Editor in *Developing SOA Applications with Oracle SOA Suite*
Creating composite sensors	Defining Composite Sensors in *Developing SOA Applications with Oracle SOA Suite*
Creating Oracle SOA Suite templates	Oracle SOA Suite Templates and Reusable Subprocesses in *Developing SOA Applications with Oracle SOA Suite*
Creating a business service in Oracle Service Bus	Developing Service Bus Applications in JDeveloper in *Developing Services with Oracle Service Bus* • Create folders • Import resources • Add business services • Create pipelines and proxies
Adding a pipeline	Working with Oracle Service Bus Pipelines in *Developing Services with Oracle Service Bus*
Creating a proxy	Creating and Configuring Proxy Services in *Developing Services with Oracle Service Bus*

4

Creating an Order Processing System

This chapter describes how Oracle SOA Suite addresses the business challenge of creating an order processing system. Overviews of how key SOA composite application components are created and address this challenge are provided, including SOA composite templates, inline BPEL subprocesses, composite sensors, and Oracle Service Bus proxy services, pipelines, and business services.
This chapter includes the following sections:

Business Challenge

Company X must design an order processing system that addresses the following business challenges:

- Many different types of clients access the system over different protocols and in different data formats, including mobile devices.
- The new order processing system must support access through REST interfaces (to prepare for a transition to an in-development mobile application).
- Existing systems must be able to place orders using XML and comma-separated value (CSV) files. These must be processed and fulfilled using the same new order provisioning system.
- The system must interface with trading partners and provide electronic data interchange (EDI) support.

Business Solution

To address these business challenges, Company X designs a business solution that uses the components described in Table 4-1.

Table 4-1 Components That Provide the Business Solution

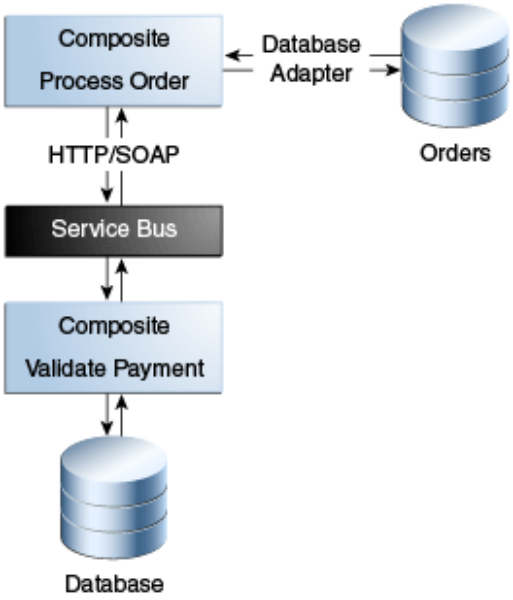
Component	How This Component Addresses The Business Challenge	Component Description
SOA composite application	A SOA composite application is designed to accept new purchase orders, authorize or deny them, and forward the authorized orders to an order fulfillment system. The composite consists of the following components, each of which is briefly described below: • SOA project template • Inline BPEL subprocess • Composite sensor	See Table 3-1 for a description of SOA composite applications.

Table 4-1 (Cont.) Components That Provide the Business Solution

Component	How This Component Addresses The Business Challenge	Component Description
SOA project template	A SOA project template is imported. The template is used to create the SOA composite application. The predefined components in the composite implement the basic scenario: • Receive an order from a web service call • Create an order number and status • Calculate the amount • Save the order in the database • Return an acknowledgement and order number to the customer The following occurs: • If the payment is denied, the order status is set to **Denied** and processing is stopped. • If the payment is authorized, the order status is set to **Authorized** and the order is sent to an order fulfillment system for processing. (Described in Fulfilling Orders .). When processing is finished, the order status is set to **ReadyForShip**.	See Table 3-1 for a description of SOA project templates.
Inline BPEL subprocess	The inline BPEL process (through use of a call activity) invokes the payment validation system in Creating a Credit Validation System to update the order status in the database based on the outcome of the payment validation.	A subprocess is a fragment of BPEL code that can be reused within a composite by separate processes. The subprocess extension provides the following benefits: • BPEL process code reusability, which reduces the need to create the same activities multiple times to perform the same tasks. • Code modularity. • Code maintenance (changes are propagated, which eliminates the need to implement updates in multiple places every time a change is necessary). • Memory footprint reduction, which can be considerable in a complex process.
Composite sensor	A composite sensor tracks the order number.	See Table 3-1 for a description of SOA composite sensors.
Oracle Service Bus proxy service, pipeline, and business service	An Oracle Service Bus makes the order processing composite available over many protocols and data formats, and validates the order.	See Table 3-1 for a description of Oracle Service Bus.

Figure 4-1 provides an overview of how this business solution is implemented.

Figure 4-1 Order Process Overview



Subsequent sections of this chapter provide more specific details about how the components in Table 4-1 are used to address the order processing business challenge.

- Creating a SOA Composite Application From a SOA Project Template
- Customizing the Contents of the SOA Project Template
- Updating Order Status with an Inline BPEL Subprocess
- Tracking the Order Number with Composite Sensors
- Updating Order Status After Payment Authorization
- Deploying and Testing in Oracle Enterprise Manager Fusion Middleware Control
- Registering the ProcessOrder Composite on Oracle Service Bus
- Testing the Pipeline Template

Creating a SOA Composite Application From a SOA Project Template

As described in Calculating Payment Status with XSLT Transformations, templates enable you to reuse existing composites, service components, and custom activities. Company X frequently has business requirements for designing SOA composite applications that accept new purchase orders, approve them, and forward them to an order fulfillment system. For this reason, Company X created a project template named **ProcessOrderTemplate** with these capabilities that can be imported into multiple applications in Oracle JDeveloper, as necessary. The template can then be customized for the business requirements of that specific project. Changes made to that specific imported template are not propagated to projects previously created using this template.

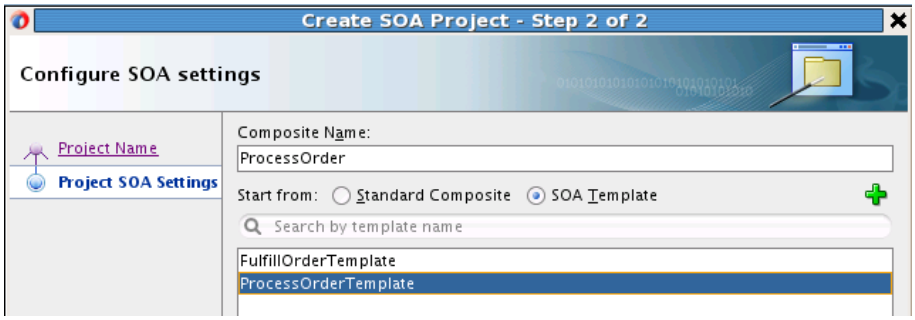
The **ProcessOrderTemplate** project template is registered in Oracle JDeveloper by selecting **Tools** > **Preferences** > **SOA** > **Templates**, and specifying the template storage location. The template is provided as a JAR file. This makes the template visible for selection in Oracle JDeveloper.

The project template consists of a number of predefined components and provides the following functionality:

- Receives an order from a SOAP web service.
- Creates an order number, sets the order date to the current date, and sets the order status to a value of **New**.
- Calculates the total order amount.
- Saves the order in the database with a status value of **New**.
- Returns an acknowledgement to the client with the order number.

Company X invokes the Create SOA Project wizard to create a new SOA project. While running the wizard, Company X selects to create a project based on a template. The project template is imported into the new application by selecting **SOA Template** in the Create SOA Project wizard, which refreshes the dialog to display existing templates for selection. **ProcessOrderTemplate** is selected, then the project name is shortened to **ProcessOrder**. Figure 4-2 provides details.

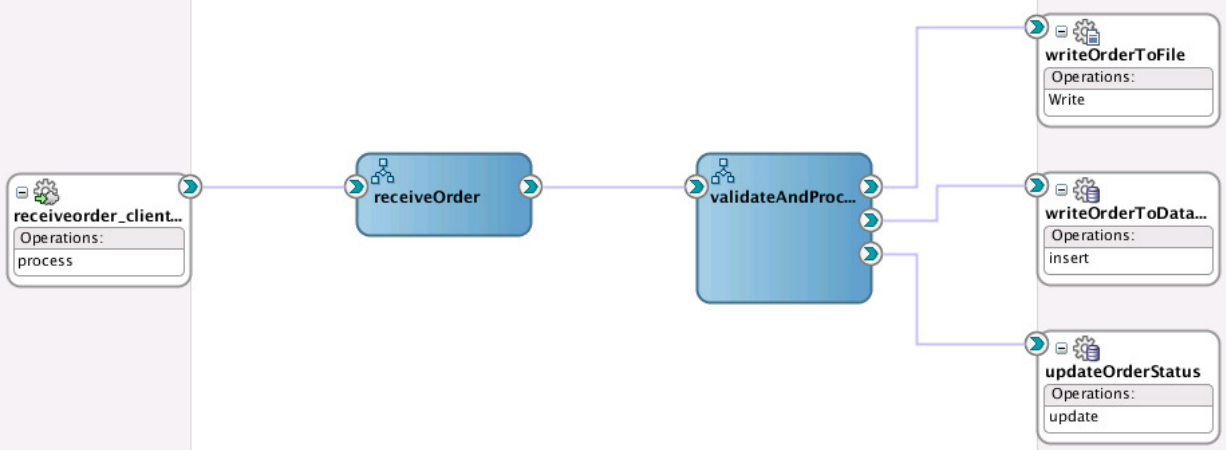
Figure 4-2 Selection of SOA Composite Template in the Create SOA Project Dialog



When imported, the project and its predefined components looks as shown in Figure 4-3.

- The **receiveOrder_client** service receives an order from a customer.
- The **receiveOrder** BPEL process service component sets an order number (which is provided back to the client) and order date, and calls the **validateAndProcessOrder** BPEL process service component.
- The **validateAndProcessOrder** BPEL process service component assigns an order to a variable and calculates the total amount of the order used to validate the payment. It invokes three partner links as part of validating and processing the order.
  - The **writeOrderToFile** file adapter reference writes the order to a file using a file adapter.
  - The **writeOrderToDatabase** reference writes the order to the database using a database adapter.
  - The **updateOrderStatus** reference updates the order status to **Denied** or **Authorized** in the database according to the value returned.

Figure 4-3 Imported Template of a SOA Composite Application



Customizing the Contents of the SOA Project Template

The **validateAndProcessOrder** BPEL process of the template assigns the order to a variable and calculates the total order amount used to validate the payment. The activities in the process shown in Figure 4-4 perform the following tasks:

- An XSLT transform activity calculates the total order amount.
- An assign activity adds the total order amount to the order message.
- A scope activity (collapsed below) includes all activities involved in updating the order status in the database and in a file.

Figure 4-4 Key Activities of the validateAndProcessOrder BPEL Process

  - If the payment is denied, the order is cancelled and the order status is updated in the database.
  - If the payment is authorized, the order status is updated in the database and file, and the order is processed. When processing is complete, the status is updated to **ReadyForShip**.

An order only requires processing if payment has first been validated. The **ProcessOrder** composite does not include this functionality. However, Company X created the

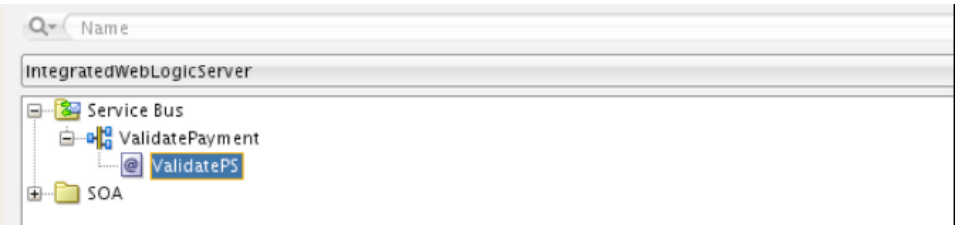
validatePayment composite in Creating a Credit Validation System. Company X customizes the imported composite template to invoke the **validatePayment** composite to validate the payment. If the payment is authorized, the **ProcessOrder** composite then processes the order. This customization to the imported template is not propagated to users of this template in other projects.

Company X customizes the **ProcessOrder** composite by dragging a SOAP web service into the **External References** swimlane to invoke the Create Web Service dialog. From this dialog, Oracle SOA Suite enables you to browse services deployed in an Oracle SOA Suite or Oracle Service Bus project on the integrated server in Oracle JDeveloper or on a remote application server. You can browse for the following:

- Select WSDL URLs.
- Read WSDLs from a file system, Oracle Metadata Services Repository (MDS Repository), UDDI registry, or web services inspection language (WSIL) file.

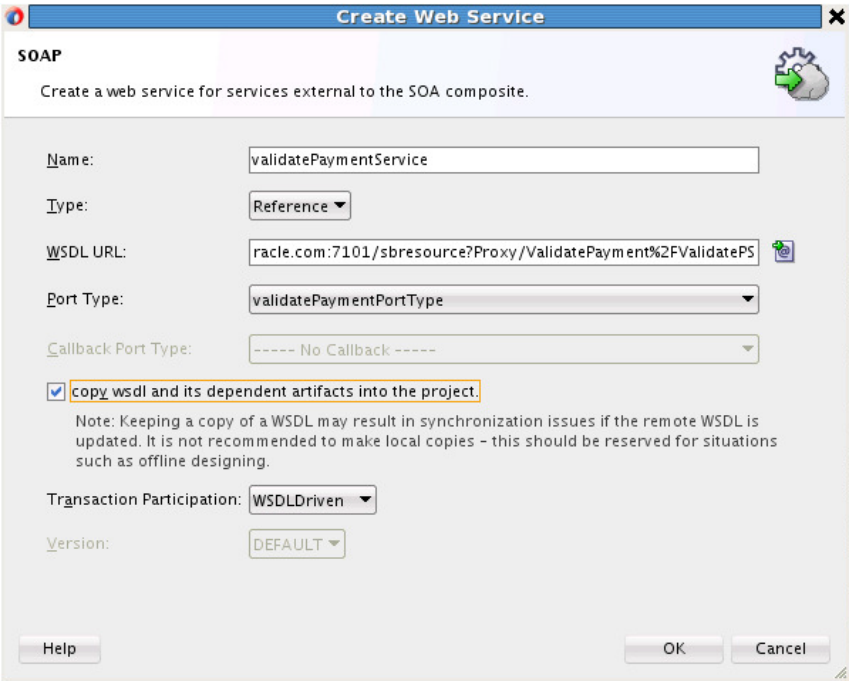
The Oracle Service Bus proxy service for **validatePayment** is selected by clicking the icon to the right of the **WSDL URL** field. Selecting this icon enables you to browse for services. The **ValidatePS** proxy service created in Registering SOA Composite Applications with Oracle Service Bus is selected, as shown in Figure 4-5.

Figure 4-5 Selection of ValidatePS Proxy in the WSDL Chooser Dialog



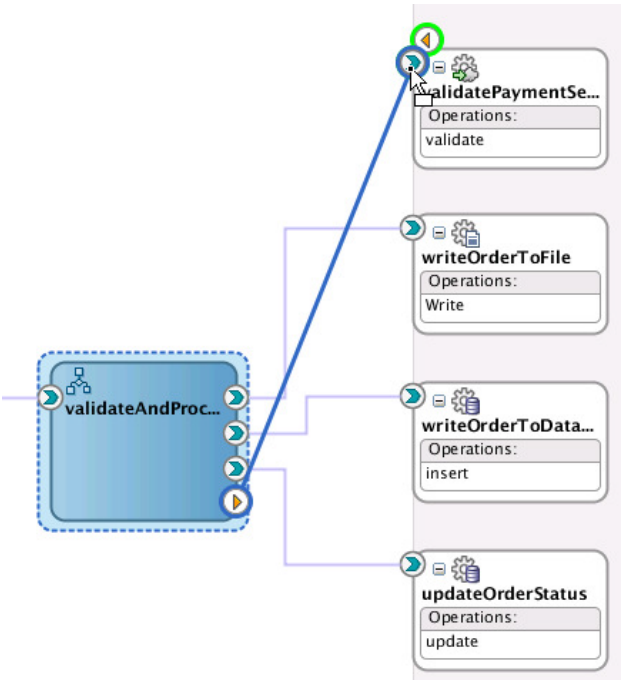
The new web service (named **validatePaymentService**) invokes the **validatePayment** proxy service defined in Registering SOA Composite Applications with Oracle Service Bus. The port type is automatically added. Figure 4-6 provides details.

Figure 4-6 Call of validatePayment Proxy WSDL File



The **validateAndProcessOrder** BPEL process is then wired to the new **validatePaymentService** SOAP web service in the SOA Composite Editor, as shown in Figure 4-7.

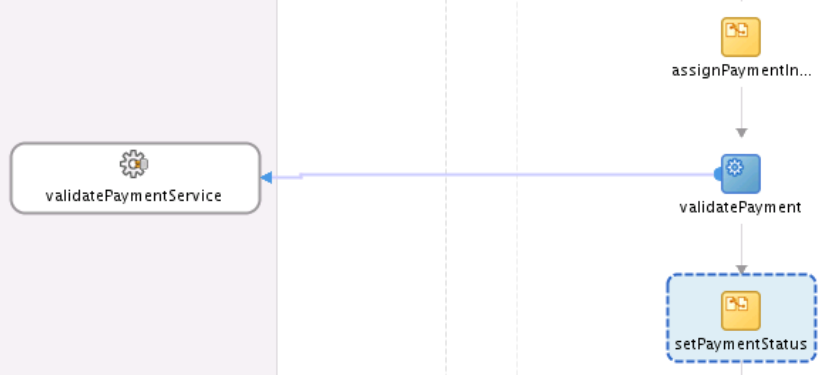
Figure 4-7 Invocation of validatePaymentService SOAP Web Service



Company X further customizes the **validateAndProcessOrder** BPEL process by adding the following:

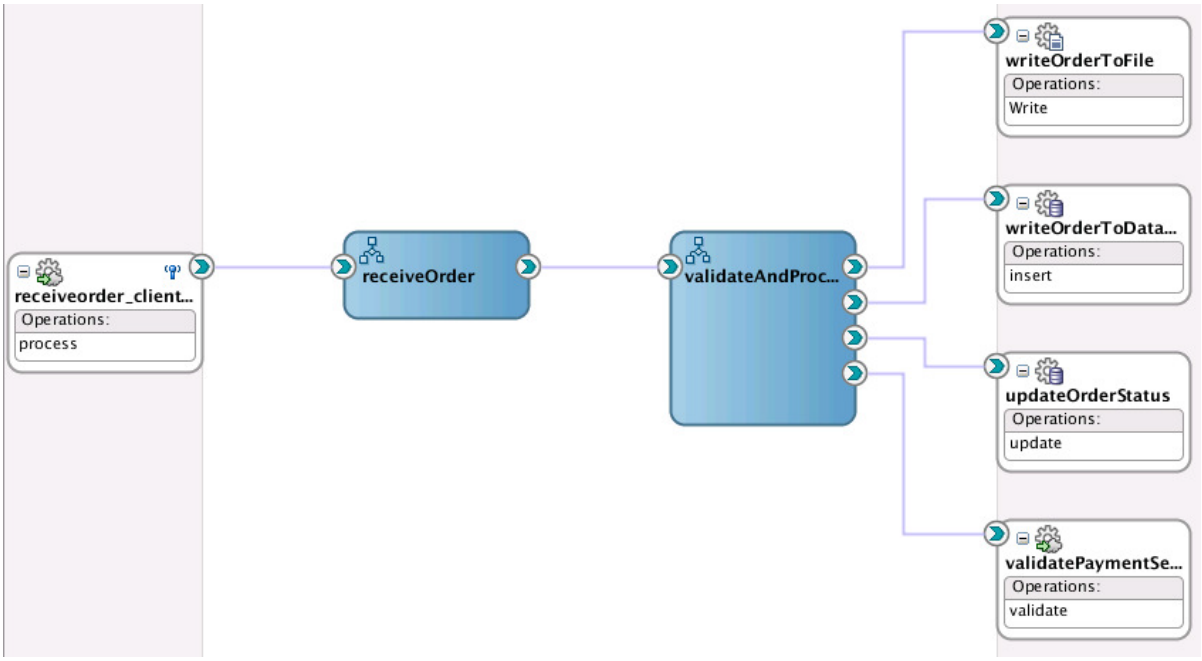
- An invoke activity (named **validatePayment**) to invoke the **validatePaymentService** partner link.
- An assign activity before the **validatePayment** invoke activity for assigning the correct values to the input variable for the web service call.
- An assign activity after the invoke activity to assign the payment status reply from the web service call to the order message. Figure 4-8 provides details.

Figure 4-8 Customizations to the validateAndProcessOrder BPEL Process of the Composite Template



The completed composite is shown in Figure 4-9.

Figure 4-9 Completed Composite



Updating Order Status with an Inline BPEL Subprocess

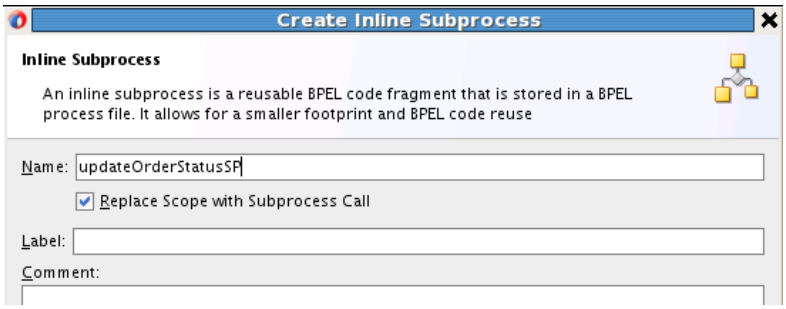
Company X has a requirement to use the order status update part of the **validateAndProcessOrder** BPEL process at least once more in the same BPEL process. One method is to create the same assign and invoke activities already used. However, this is an error prone process and every time a change is necessary, it must be made in all those places. To avoid this, Company X uses a BPEL subprocesses. There are two types:

- Standalone: A fragment of a BPEL process, which includes a number of activities to reuse. Standalone subprocess do not have an interface and are only called from another BPEL process. A standalone process can have partner links across a number of other BPEL processes.
- Inline: For groups of activities that are reused within a single BPEL process. An inline process is part of the parent BPEL process code and is not visible in the composite view. You use a call activity for inline subprocesses.

  The inline subprocess is ideal for Company X's business requirements.

Within the **validateAndProcessOrder** BPEL process, the scope activity responsible for updating the order status, named **updateOrderStatusScope**, is right-clicked, and **Convert to a Subprocess** is selected. This invokes the Create Inline Subprocess dialog. Company X renames the subprocess and selects to automatically replace the scope activity with a subprocess call activity. Figure 4-10 provides details.

Figure 4-10 Create Inline Subprocess



This converts the scope activity into a call activity. A call activity executes referenced subprocess code in standalone and inline subprocesses. The call activity is also added to the **Subprocesses** part of the Components window. You can drag and drop this call activity as needed into other locations in the BPEL process. Figure 4-11 provides details.

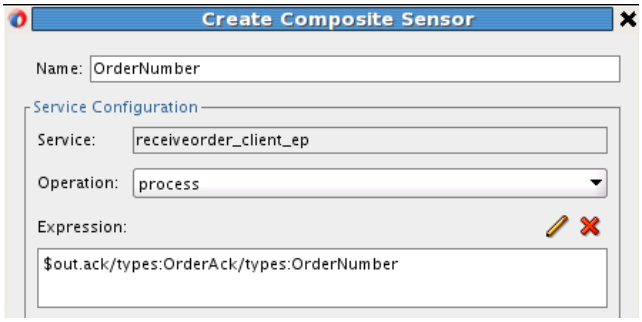
Figure 4-11 Call Activity in the BPEL Process and in the Components Window

Tracking the Order Number with Composite Sensors

Company X added a composite sensor for tracking the status of order payments in Tracking Payment Status with Composite Sensors.

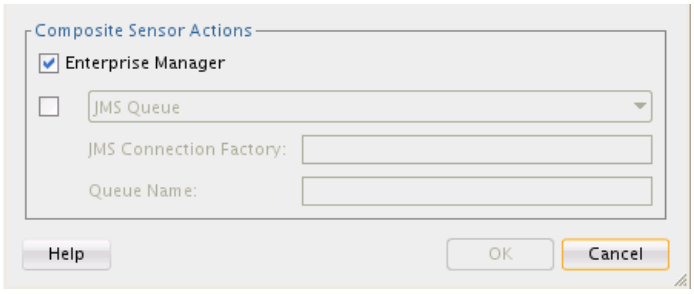
Company X now has an additional requirement for a composite sensor to track the order number. The SOAP web service **receiveorder_client** included in the imported composite template shown in Figure 4-3 returns the **orderNumber** in the order acknowledgement message. Company X defines a composite sensor on this service that includes an XPath expression to track the order number, as shown in the Create Composite Sensor dialog in Figure 4-12.

Figure 4-12 Expression Defined on Composite Sensor



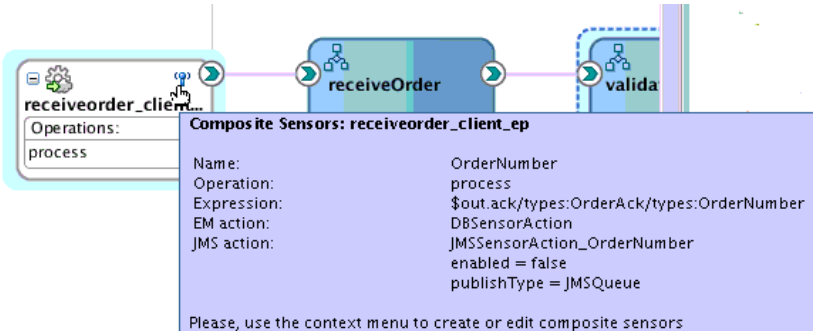
The **Enterprise Manager** check box of the Create Composite Sensor dialog is also selected, as shown in Figure 4-13. This enables you to track composite sensor names and values (for example, **OrderNumber=1234**) on the Flow Instances page or the Flow Trace page for a specific business flow instance in Oracle Enterprise Manager Fusion Middleware Control. Oracle Enterprise Manager Fusion Middleware Control is a web browser-based, graphical user interface that you use to monitor and administer your deployed composites.

Figure 4-13 Enterprise Manager Check Box for Composite Sensors



The composite sensor definition is displayed by placing the cursor over the icon on the **receiveorder_client** SOAP web service binding component. Figure 4-14 provides details.

Figure 4-14 Composite Sensor Definition



Updating Order Status After Payment Authorization

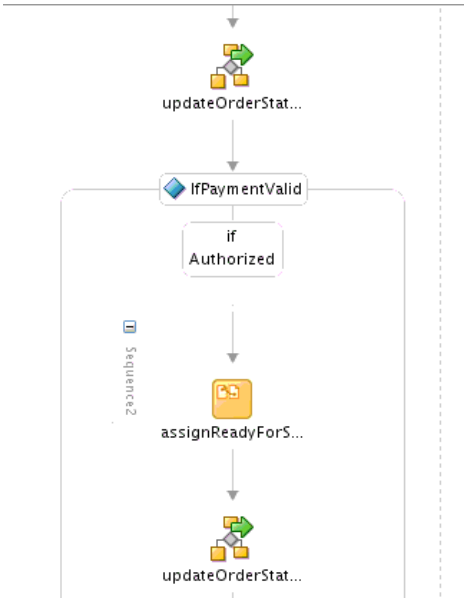
If the payment is valid, the order status is set to **ReadyForShip** in the database. This status update triggers the order fulfillment process described in Fulfilling Orders .

Company X further customizes the **validateAndProcessOrder** BPEL process by adding an if activity. An if activity defines conditional behavior for specific activities to decide between the execution of two or more branches. Only one activity is selected for execution from a set of branches. The if activity for this business scenario consists of the following branches:

- If the payment is authorized, the order continues. An assign activity in the if branch updates the order status to **ReadyForShip** in the database.
- If the payment is denied, processing ends and an email is sent to the customer informing them about the unauthorized payment.

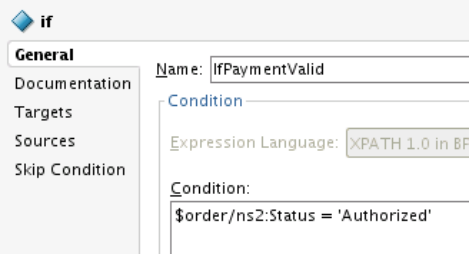
Figure 4-15 provides details. The if activity is added below the **updateOrderStatusSP** call activity that was created in Updating Order Status with an Inline BPEL Subprocess.

Figure 4-15 If Activity



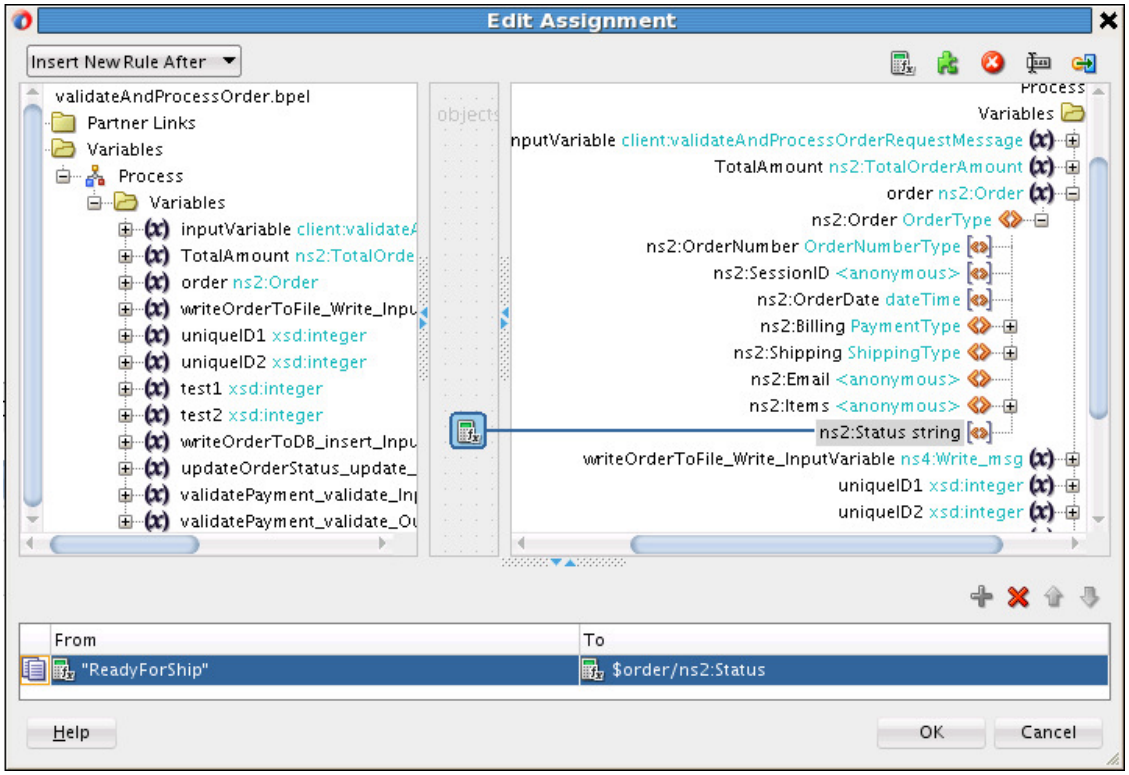
An expression is defined on the **if** branch if the payment is authorized, as shown in Figure 4-16.

Figure 4-16 If Branch Authorizes Payment



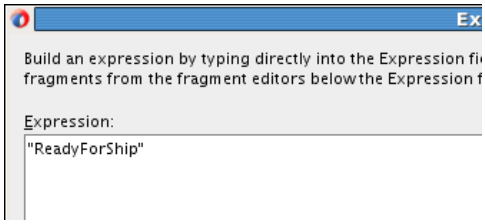
If payment is authorized, the processing of the order is complete. An assign activity is added to the **if** branch to update the order status to **ReadyForShip** in the database. The contents of the copy rule in the assign activity are shown in Figure 4-17.

Figure 4-17 Assign Activity



The XPath expression contents in the copy rule assign activity are shown in Figure 4-18.

Figure 4-18 ReadyForShip Expression



An **else** branch is not necessary and is deleted because order processing stops if the payment has been denied.

Deploying and Testing in Oracle Enterprise Manager Fusion Middleware Control

Company X deploys and creates a business flow instance of the project. In the Flow Trace page in Oracle Enterprise Manager Fusion Middleware Control, the two composite sensor names and values are displayed. Payment has been authorized, and the project is sent to the order fulfillment system. Figure 4-19 provides details.

Figure 4-19 Composite Sensor Names and Values on Flow Trace Page

Sensor Name	Value	Location	Composite
OrderNumber	**201342093899**	receiveorder_client_e	ProcessOrder [1.0]
PaymentStatus	**Authorized**	validatepaymentproc	ValidatePayment [1.0]

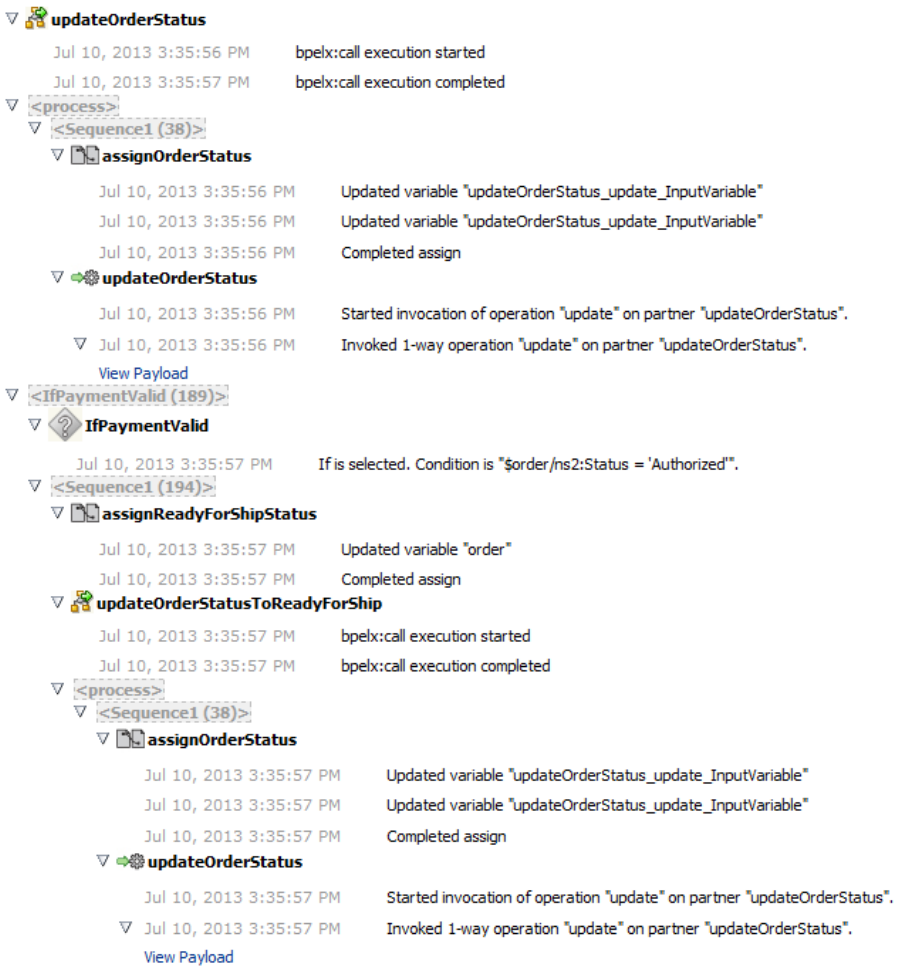
Trace

Actions ▾ View ▾ Show Instance IDs

Instance	Type	Usage	State	Time	Composite
receiveorder_client_ep	Service	Service	Completed	Apr 20, 2013 9:38:09 AM	ProcessOrder [1.0]
receiveOrder	BPEL		Completed	Apr 20, 2013 9:38:09 AM	ProcessOrder [1.0]
validateAndProcessOrder	BPEL		Completed	Apr 20, 2013 9:38:09 AM	ProcessOrder [1.0]
writeOrderToDatabase	Reference	Reference	Completed	Apr 20, 2013 9:38:10 AM	ProcessOrder [1.0]
writeOrderToFile	Reference	Reference	Completed	Apr 20, 2013 9:38:11 AM	ProcessOrder [1.0]
validatePaymentService	Reference	Reference	Completed	Apr 20, 2013 9:38:11 AM	ProcessOrder [1.0]
OSB	OSB			Apr 20, 2013 9:38:13 AM	
validatepaymentprocess_client_ep	Service	Service	Completed	Apr 20, 2013 9:38:13 AM	ValidatePayment [1.0]
validatePaymentProcess	BPEL		Completed	Apr 20, 2013 9:38:13 AM	ValidatePayment [1.0]
getPaymentInformation	Reference	Reference	Completed	Apr 20, 2013 9:38:13 AM	ValidatePayment [1.0]
updateOrderStatus	Reference	Reference	Completed	Apr 20, 2013 9:38:14 AM	ProcessOrder [1.0]
updateOrderStatus	Reference	Reference	Completed	Apr 20, 2013 9:38:14 AM	ProcessOrder [1.0]

The audit trail indicates that the order is marked as **Authorized** and **ReadyForShip**. Figure 4-20 provides details.

Figure 4-20 Flow of Business Flow Instance



Registering the ProcessOrder Composite on Oracle Service Bus

As with the **validatePayment** composite in Creating a Credit Validation System, Company X uses Oracle Service Bus to register the **ProcessOrder** composite to make it available to external customers.

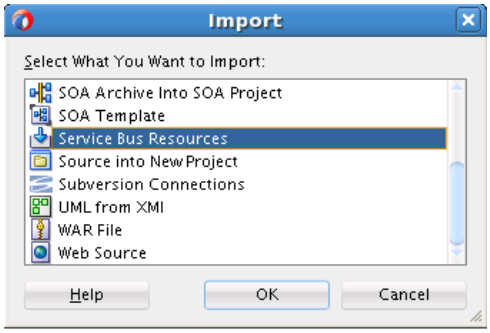
Oracle Service Bus makes the **ProcessOrder** composite available over different protocols and data formats without disrupting the core business logic in the composite. Oracle Service Bus also validates the order data and performs auditing.

Company X updates the Oracle Service Bus application created in Registering SOA Composite Applications with Oracle Service Bus to include additional projects. These projects use a pipeline template.

Pipeline templates are used to design prototype message flows for proxy services. A pipeline template defines the general shape or pattern of the message flow. Concrete pipelines can then be generated out of the pipeline template. All concrete pipelines use the message flow defined by the pipeline template with designated places where custom logic can be inserted.

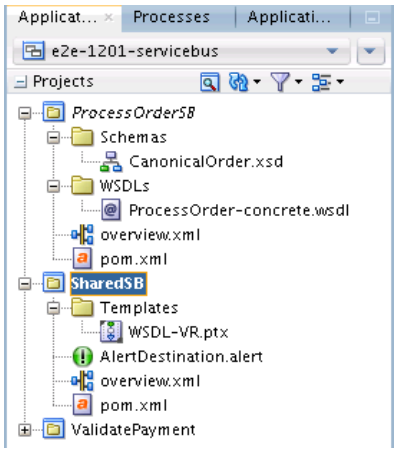
Company X imports the template by right-clicking the Oracle Service Bus application, selecting **Import**, and selecting **Service Bus Resources**. Figure 4-21 provides details.

Figure 4-21 Oracle Service Bus Resources Selection



This invokes the Import Service Bus Resources dialog for selecting the pipeline template to use. When complete, the imported template is displayed in the Applications window, as shown in Figure 4-22.

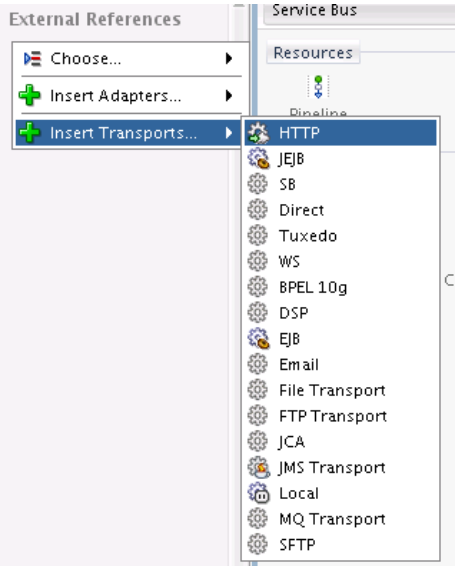
Figure 4-22 Imported Oracle Service Bus Pipeline Template in Applications Window



Registering the ProcessOrder Composite as a Business Service

Company X registers the composite as a business service by right-clicking in the **External References** column and selecting **Insert Transports** > **HTTP**, as shown in Figure 4-23.

Figure 4-23 Business Service Creation



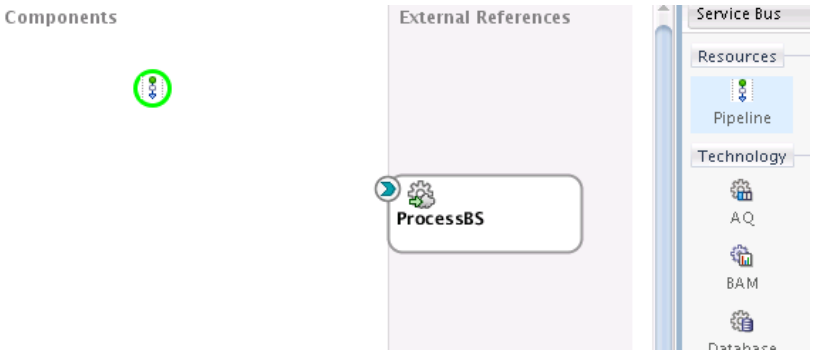
This action invokes the Create Business Service wizard in which Company X sets the following:

- HTTP as the transport type
- **ProcessOrder** concrete WSDL as the service type
- The WSDL file
- The composite as the endpoint URI. Oracle Service Bus also enables you to have multiple endpoints for your business service to support application load balancing and failover.

Creating a New Pipeline with a Proxy Using the Pipeline Template

Company X then creates a pipeline template (**SharedSB**) by dragging a **Pipeline** icon from the Component window into the **Components** section of Oracle JDeveloper, as shown in Figure 4-24.

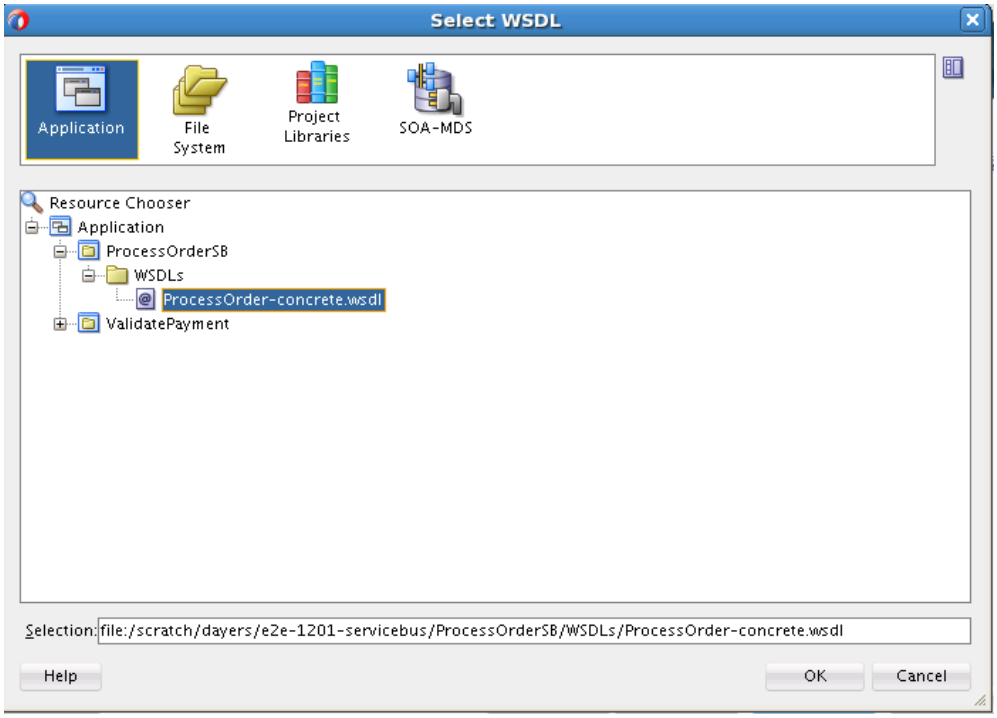
Figure 4-24 Pipeline Template



This invokes the Create Pipeline Service dialog for selecting the imported pipeline template to use. During configuration, you select the template for WSDL-based services, the specific

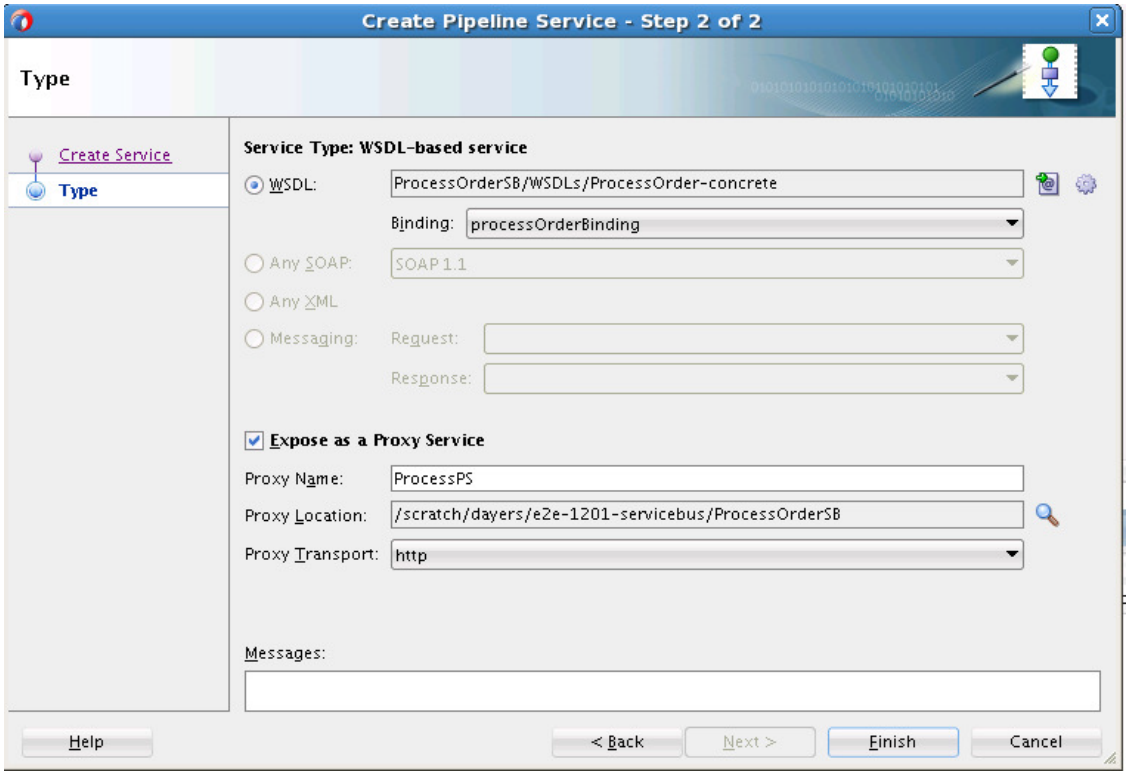
WSDL to use for the pipeline (shown in Figure 4-25), and to expose the pipeline as a proxy service.

Figure 4-25 Select WSDL Dialog



When configuration is complete, the Create Pipeline Service dialog looks as shown in Figure 4-26.

Figure 4-26 Completed Create Pipeline Service Dialog

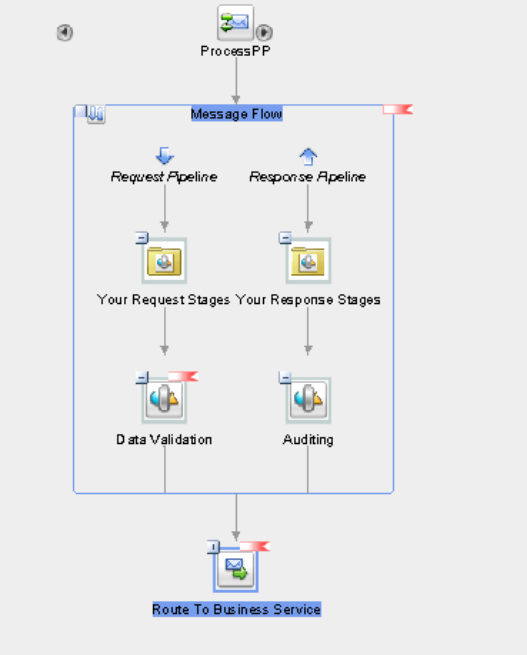


The pipeline is displayed in the application, as shown in Figure 4-27. The yellow icon in the pipeline indicates that it requires additional configuration before it is fully implemented.

Figure 4-27 Pipeline in Application

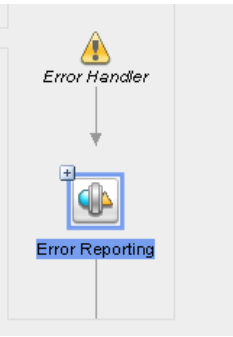
Double-clicking **ProcessPP** displays it for editing, and includes some features partially preconfigured (for example, data validation, routing, reporting, error condition alerts, and error handling). Red flags indicate areas that require additional configuration. The **Your Request Stages** message indicates areas in which the pipeline provides placeholder information for Company X to customize (such as transformations and message enrichment). Figure 4-28 provides details.

Figure 4-28 Preconfigured Pipeline Template



The template also enables you to define an error handler for pipelines, as shown in Figure 4-29. The predefined error handler reports error back to the caller with details and send an alert to Oracle Enterprise Manager Fusion Middleware Control indicating that something is wrong with processing.

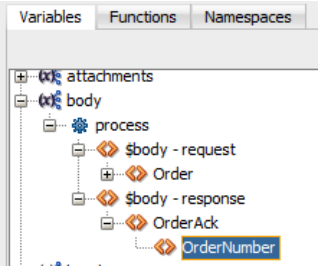
Figure 4-29 Error Handling in the Pipelines Template



Company X configures data validation by opening the **Data Validation** stage and selecting the **Validate** action. This part enables Company X to validate the incoming payload against the **Order** schema element type expected by the **ProcessOrder** composite. Validating in Oracle Service Bus saves resources in the back end that are processing good orders. Company X only needs to select the **Order** type against which to validate, since all other details are preconfigured.

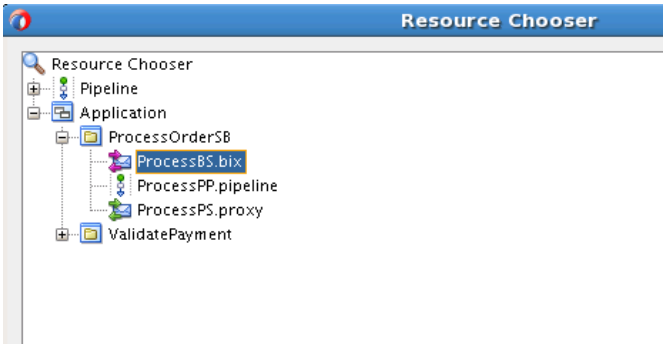
Company X configures the **Report** action in the **Auditing** stage of the response flow shown in Figure 4-28 to report on the **Order** number returned from the composite, as shown in Figure 4-30. The template already saves a copy of the incoming order in case of an error and reports it to Oracle Enterprise Manager Fusion Middleware Control.

Figure 4-30 OrderNumber Selection



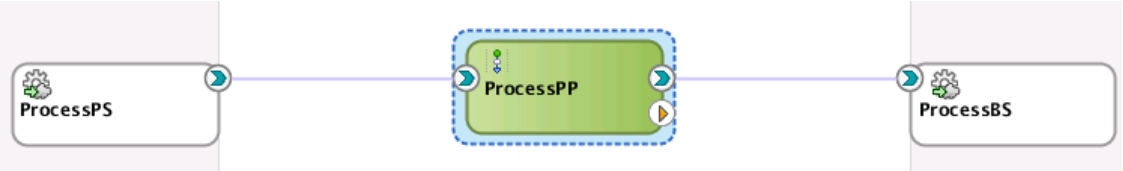
Company X also configures the routing node by double-clicking the **Routing** icon in the template in Figure 4-28, selecting the **Routing** tab in the Property Inspector, and selecting the **ProcessBS** business service. Figure 4-31 provides details.

Figure 4-31 Business Service Selection



When configuration is complete, the Oracle Service Bus application looks as shown in Figure 4-32.

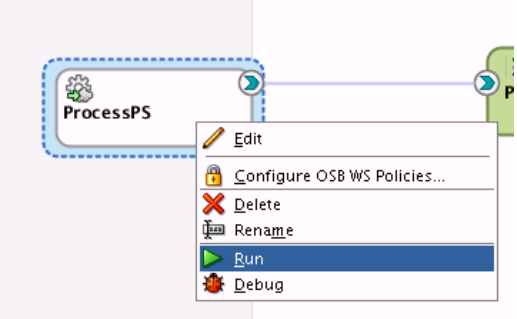
Figure 4-32 Oracle Service Bus Application



Testing the Pipeline Template

Company X tests by right-clicking the **ProcessPS** service in the left swim lane and selecting **Run**, as shown in Figure 4-33. Good and bad orders are sent during the test phase.

Figure 4-33 Testing the Pipeline Template



Related Documentation

Table 4-2 provides references to documentation that more specifically describes the components and features described in this chapter.

Table 4-2 Related Documentation

For Information About...	See...
Creating a SOA composite application	Creating a SOA Application in *Developing SOA Applications with Oracle SOA Suite*
Creating and using composite templates and inline subprocesses	Oracle SOA Suite Templates and Reusable Subprocesses in *Developing SOA Applications with Oracle SOA Suite*
Creating composite sensors	Defining Composite Sensors in *Developing SOA Applications with Oracle SOA Suite*
Creating a business service in Oracle Service Bus	Creating and Configuring Business Services in *Developing Services with Oracle Service Bus*
Using pipeline templates	Working with Pipeline Templates in *Developing Services with Oracle Service Bus*

5

Adding New Ordering Channels with Oracle Service Bus

This chapter describes how Oracle SOA Suite addresses the business challenge of adding new ordering channels. Overviews of how key components are used to address this challenge are provided, including file-based proxies in Oracle Service Bus, the Native Format Builder Wizard, Oracle Enterprise Manager Fusion Middleware Control, and the Oracle Service Bus debugger.
This chapter includes the following sections:

Business Challenge

Company X has a requirement for legacy systems to place orders in common delimited formats.without impacting the back end application. Protocol and data translation must be provided to minimize disruptions to the back end business logic of the **ProcessOrder** composite designed in Creating an Order Processing System.

Business Solution

To address these business challenges, Company X designs a business solution that uses the components described in Table 5-1.

Table 5-1 Components That Provide the Business Solution

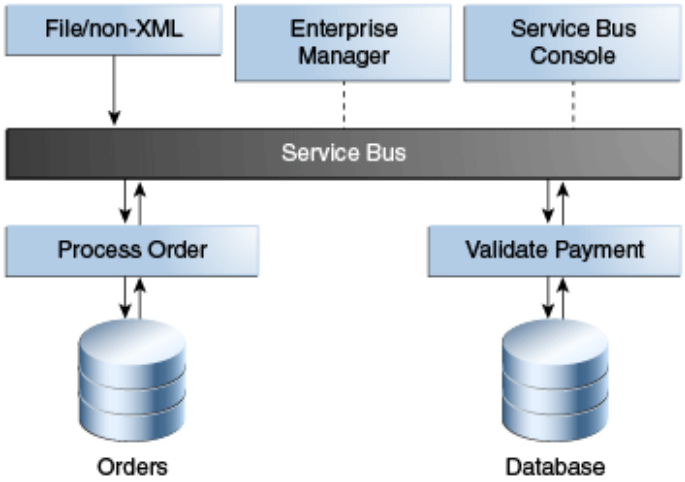
Component	How This Component Addresses The Business Challenge	Component Description
Oracle Service Bus file-based proxy	An Oracle Service Bus pipeline connects a proxy service to a file ordering channel. The proxy handles incoming customer orders by file.	A pipeline validates the application before invocation. A proxy enables customers to invoke the composite through a proxy instead of connecting directly to the composite.
Native Format Builder Wizard	The file adapter transforms a comma-delimited order to an XML schema file for the order processing composite to use.	The Native Format Builder Wizard enables you to create a native schema file from a variety of file formats, such as delimited, fixed-length, complex type, data type description (DTD), and Cobol Copybook.
Oracle Service Bus Debugger	A breakpoint debugger set on the Oracle Service Bus pipeline tests and debugs the end-to-end application.	The Oracle Service Bus debugger enables you to set breakpoints directly in Oracle JDeveloper for troubleshooting on pipelines and split-joins.

Table 5-1 (Cont.) Components That Provide the Business Solution

Component	How This Component Addresses The Business Challenge	Component Description
Oracle Enterprise Manager Fusion Middleware Control	Oracle Enterprise Manager enables you to monitor and administer an Oracle Service Bus application.	Oracle Enterprise Manager Fusion Middleware Control provides a web browser-based, graphical user interface for monitoring and administering deployed applications.

Figure 5-1 provides an overview of how this business solution is implemented.

Figure 5-1 Ordering Channels



Subsequent sections of this chapter provide more specific details about how the components in Table 5-1 are used to address the new order channel business challenge.

- Adding a File-Based Proxy to the Oracle Service Bus Pipeline
- Debugging Components with the Oracle Service Bus Debugger
- Monitoring Oracle Service Bus in Oracle Enterprise Manager Fusion Middleware Control

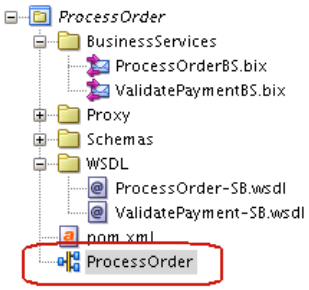
Adding a File-Based Proxy to the Oracle Service Bus Pipeline

Company X's customer base includes legacy companies that are unable to submit their orders in XML format. They instead use a comma-delimited format. Company X must be able to transform these comma-delimited orders into a valid XML format.

Oracle SOA Suite includes an Adapter Configuration Wizard that guides you through integrating applications with file systems, database tables, database queues, FTP servers, Java Message Services (JMS), IBM WebSphere MQ, and other systems. In 12*c*, the Adapter Configuration Wizard is available in both Oracle SOA Suite and Oracle Service Bus.

Company X opens the **ProcessOrder** project in the Oracle Service Bus application in the Applications window in Oracle JDeveloper. Figure 5-2 provides details.

Figure 5-2 File Adapter Icon in Components Window

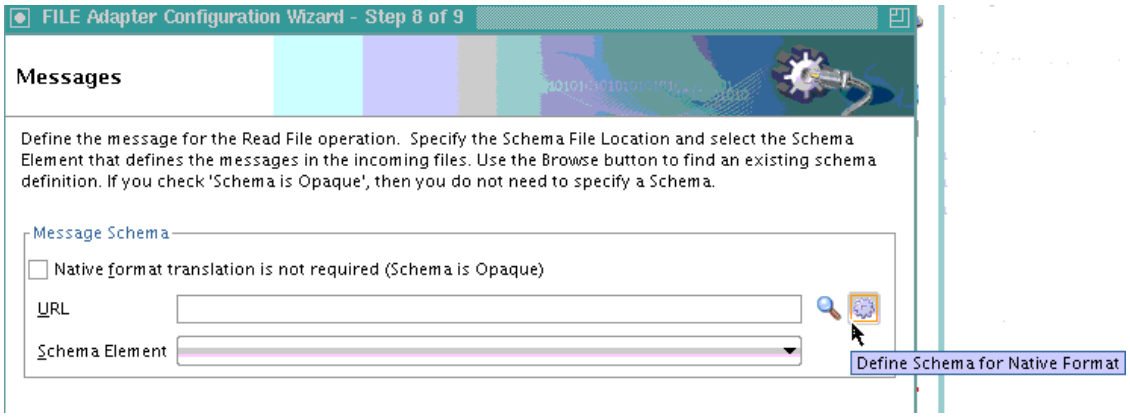


Company X then drags a **File** icon from the Components window to the **Exposed Services** swimlane. This invokes the Adapter Configuration Wizard for adding a new file-based proxy for the **ProcessOrder** composite.

The file adapter lets Oracle Service Bus business and proxy services exchange (read and write) files on local file systems. The file contents can be in both XML and non-XML data formats. For this scenario, the file adapter is configured to read comma-separated value (`.csv`) files from a specified directory that is polled every minute. On the last page of the wizard, Company X selects the option to create a valid XML schema file from the native comma-delimited format submitted by customers. This selection invokes the Native Format Builder Wizard. Figure 5-3 provides details.

The Native Format Builder Wizard enables you to create a native schema file from a variety of file formats, such as delimited, fixed-length, complex type, data type description (DTD), and Cobol Copybook. Native schema is an XML schema definition with annotations and additional attributes that can be used to translate native format files to XML, and vice versa. This native schema enables the Adapter Configuration Wizard to create a WSDL file for the adapter to communicate with the application.

Figure 5-3 Selection of Option to Define a Schema for a Native Format



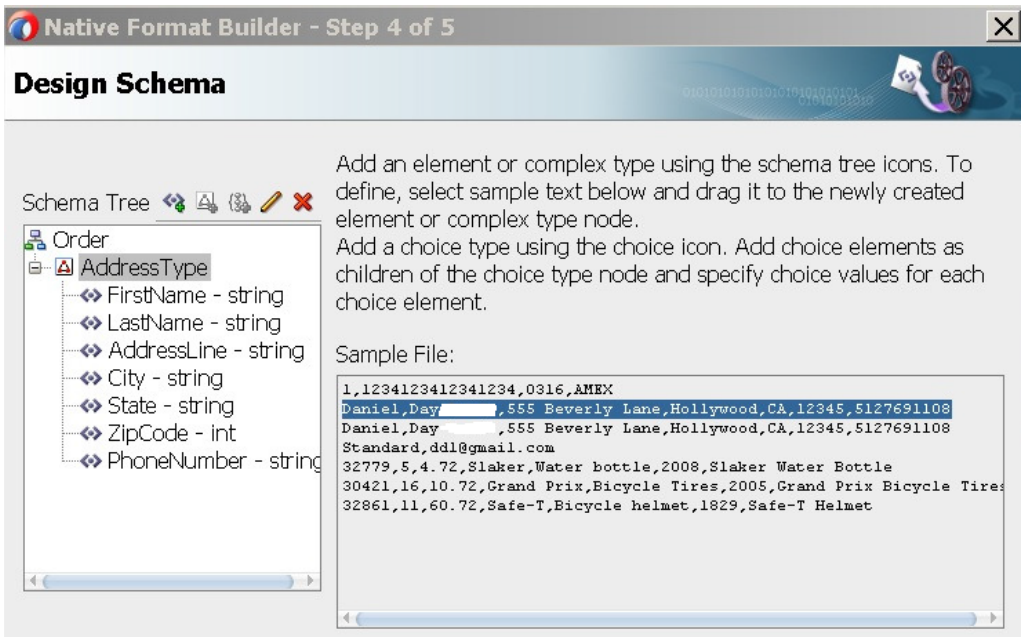
For this scenario, Company X configures the Native Format Builder Wizard pages as follows:

- Selects **Complex Type (Contains records whose fields may themselves be records having multiple delimiter types)** as the native message format from which to create a schema.
- Specifies the native format file and the root element (`Order`).

- Specifies the complex types for data corresponding to address, billing, shipping, and item.
- Specifies the delimiter for data (comma).
- Specifies the schema elements corresponding to the fields in the data file.
- Renames the automatically generated elements C1,C2.....C7 to element names that are expected in the generated XML file.

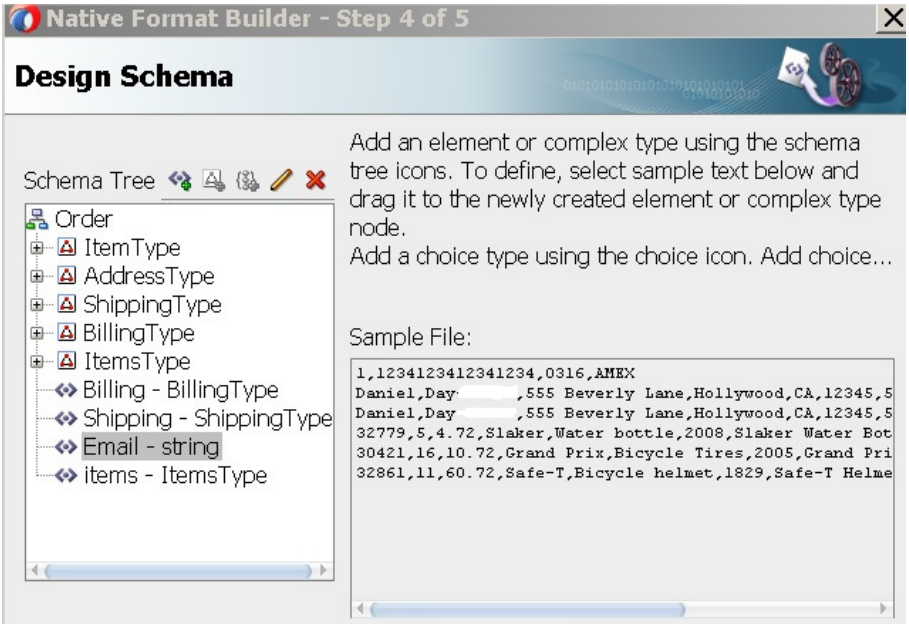
Figure 5-4 shows the complex type **AddressType** with defined schema elements for **FirstName**, **LastName**, **AddressLine**, **City**, **State**, **ZipCode**, and **PhoneNumber**. The comma-delimited file is shown on the right side of the page.

Figure 5-4 Designing of Complex Types in the Schema File



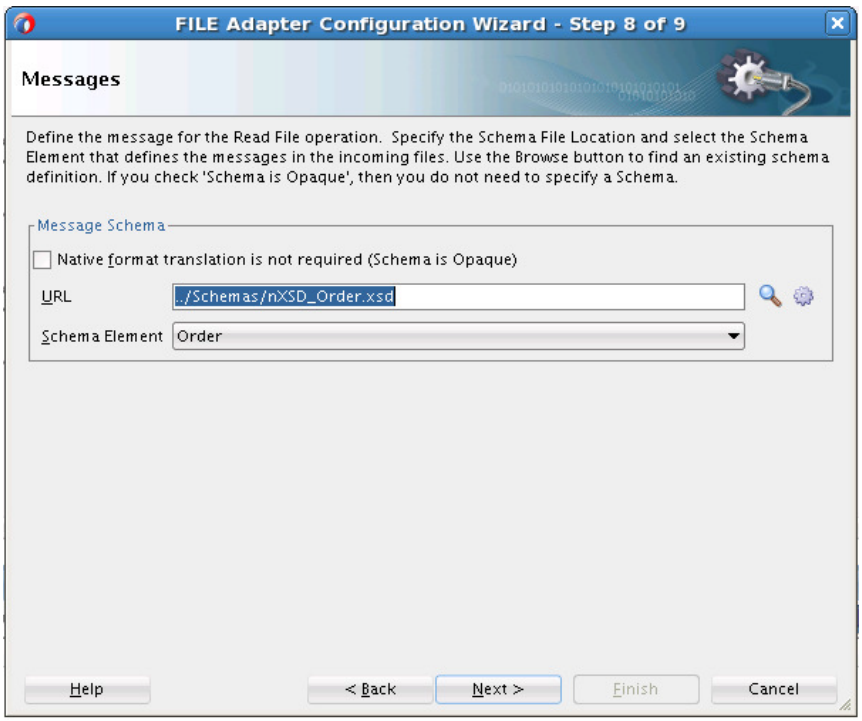
The configuration process is repeated for all required complex types, as shown in Figure 5-5.

Figure 5-5 Completion of Design of Complex Types in the Schema File



When complete, a schema file is created. Figure 5-6 provides details.

Figure 5-6 Schema File and Schema Element Creation



Company X then wires the newly created **ProcessPS_File** proxy to the **ProcessPP** pipeline. Figure 5-7 provides details.

Figure 5-7 File Proxy Wired to ProcessPP Pipeline

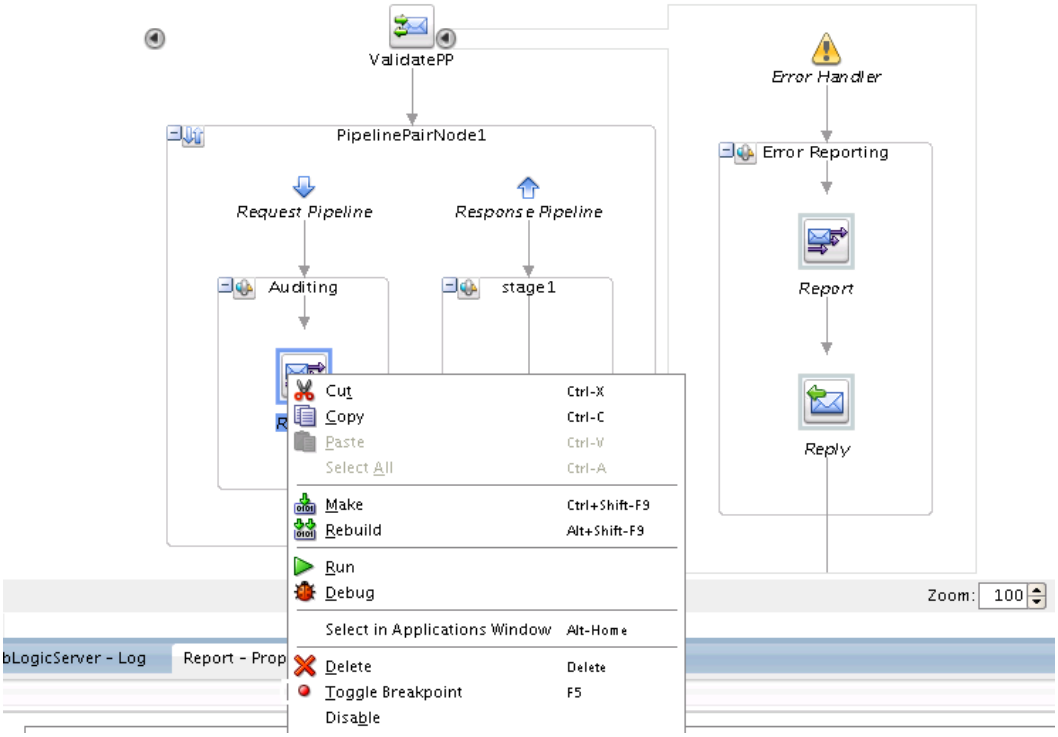
Debugging Components with the Oracle Service Bus Debugger

Oracle JDeveloper provides a comprehensive debugger for assessing and repairing Oracle Service Bus project components. The debugger reduces the development cycle by providing a troubleshooting environment directly in Oracle JDeveloper. This means you do not need to build an Oracle Service Bus application in Oracle JDeveloper, run it, launch a console to test or view audit trails and flow traces, and then return to Oracle JDeveloper to fix any issues and repeat these steps. Instead, you can set breakpoints directly in Oracle JDeveloper for troubleshooting on pipelines and split-joins.

Company X navigates to the **validatePayment** pipeline to debug the pipeline in Oracle JDeveloper. Company X selects the **Reporting** action in the pipeline editor and right-clicks and selects **Toggle Breakpoint** from the menu.

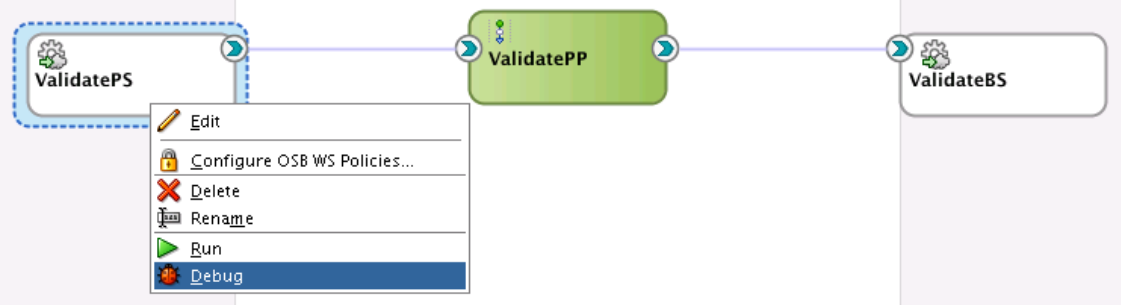
Figure 5-8 provides details.

Figure 5-8 Set Breakpoints



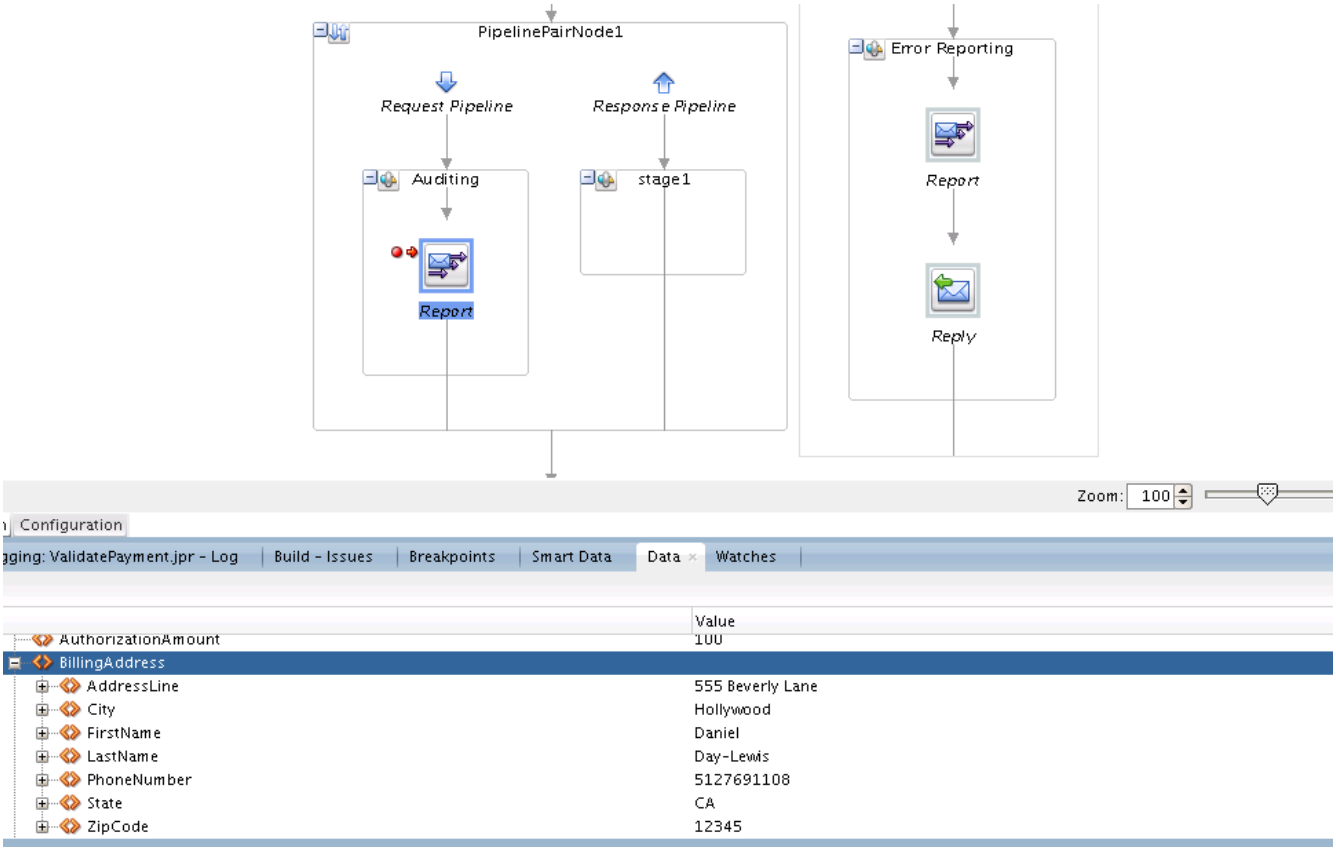
With a breakpoint set, Company X invokes the debugger by right-clicking the proxy and selecting **Run**. Figure 5-9 provides details.

Figure 5-9 Start Debugging of the Oracle Service Bus Proxy



This enables Company X to enter sample data in the test console, review the variables, step through the pipeline in debug mode, and change data, as necessary. Figure 5-10 provides details.

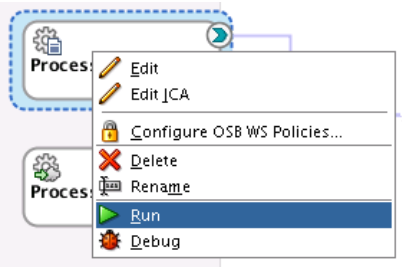
Figure 5-10 Debug of Oracle Service Bus Pipeline



Monitoring Oracle Service Bus in Oracle Enterprise Manager Fusion Middleware Control

Company X deploys the Oracle Service Bus application, as shown in Figure 5-11.

Figure 5-11 Deployment of Oracle Service Bus Application

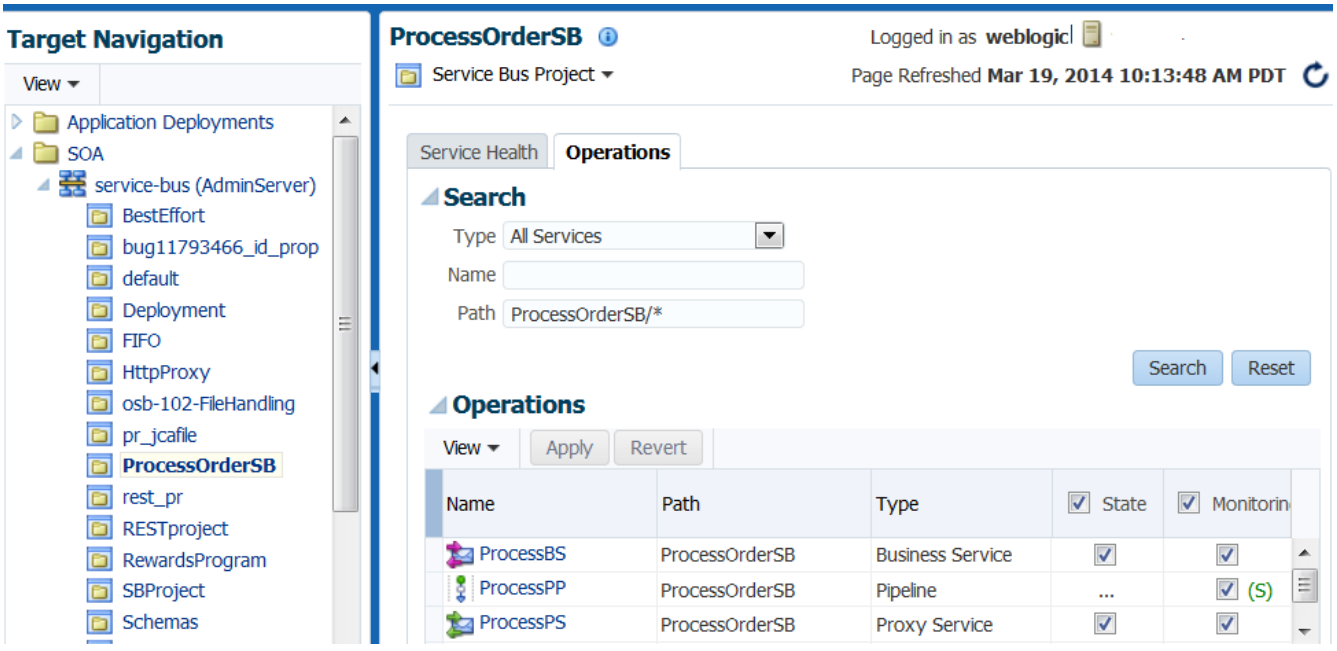


Company X monitors Oracle Service Bus from Oracle Enterprise Manager Fusion Middleware Control. Oracle Enterprise Manager Fusion Middleware Control is a web browser-based, graphical user interface that you use to monitor and administer deployed applications.

Starting with 12*c*, the administration of Oracle Service Bus and Oracle SOA Suite is performed from a single, unified Oracle Enterprise Manager Fusion Middleware Control.

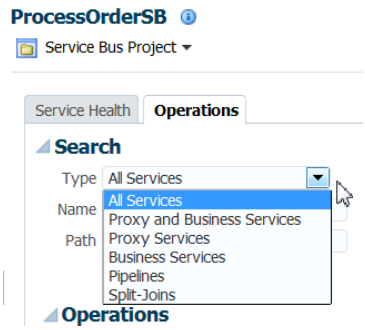
Services are not displayed by default to optimize performance. Therefore, Company X selects **ProcessOrderSB** in the navigator, clicks the **Operations** tab, and clicks **Search** to display all the services in **ProcessOrderSB**. Figure 5-12 provides details.

Figure 5-12 ProcessOrderSB Search



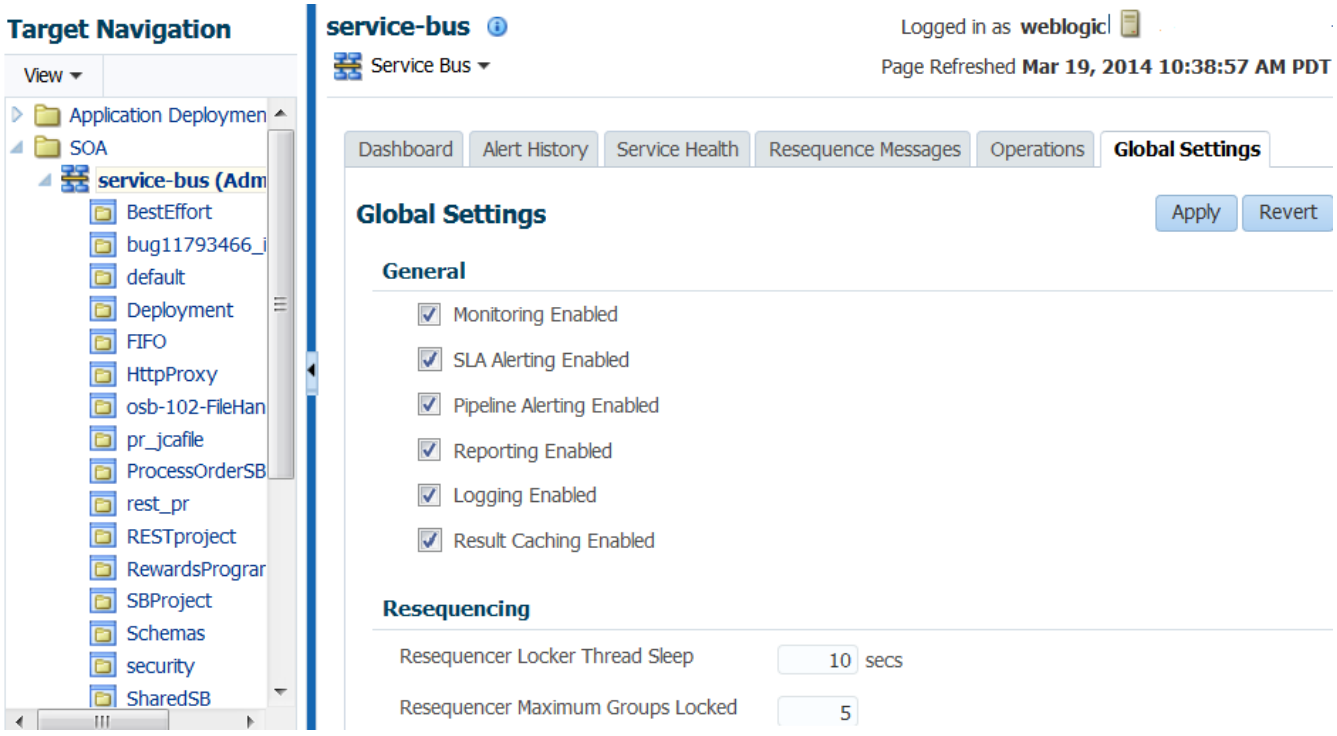
The **Type** list enables you to filter the display of details, as shown in Figure 5-13.

Figure 5-13 Filter Display of Details



Company X uses the Oracle Service Bus home page to monitor additional details under each of the tabs displayed at the top of the page. Figure 5-14 provides details of the **Global Settings** tab.

Figure 5-14 Oracle Service Bus Home Page



Related Documentation

Table 5-2 provides references to documentation that more specifically describes the components and features described in this chapter.

Table 5-2 Related Documentation

For Information About...	See...
Configuring the file adapter with the Adapter Configuration Wizard	Oracle JCA Adapter for Files/FTP in *Understanding Technology Adapters*
Creating the native schema file with the Native Format Builder Wizard	Native Format Builder Wizard in *Understanding Technology Adapters*
Monitoring Oracle Service Bus in Oracle Enterprise Manager Fusion Middleware Control	Getting Started with Oracle Service Bus Administration in *Administering Oracle Service Bus*
Debugging Oracle Service Bus applications in Oracle JDeveloper	Debugging Oracle Service Bus Applications in *Developing Services with Oracle Service Bus*

6

Packing and Shipping Orders

This chapter describes how Oracle SOA Suite addresses the business challenge of packing and shipping orders. Overviews of how key SOA composite application components address this challenge are provided, including a REST service, BPEL process, HTTP Analyzer, BPEL process component template, standalone BPEL subprocess, composite sensor, and Oracle User Messaging Service adapter.
This chapter includes the following sections:

Business Challenge

Company X has a requirement for assigning preferred shipping providers to a specific shipping method. This method is calculated based on the shipping speed the customer selected when placing the order and the shipping state (in the address). Once the order has been shipped, a notification must be sent to the customer confirming the shipping provider. The order status must also be updated to shipped.

Business Solution

To address this business challenge, Company X designs a business solution that uses the components described in Table 6-1.

Table 6-1 Components That Provide the Business Solution

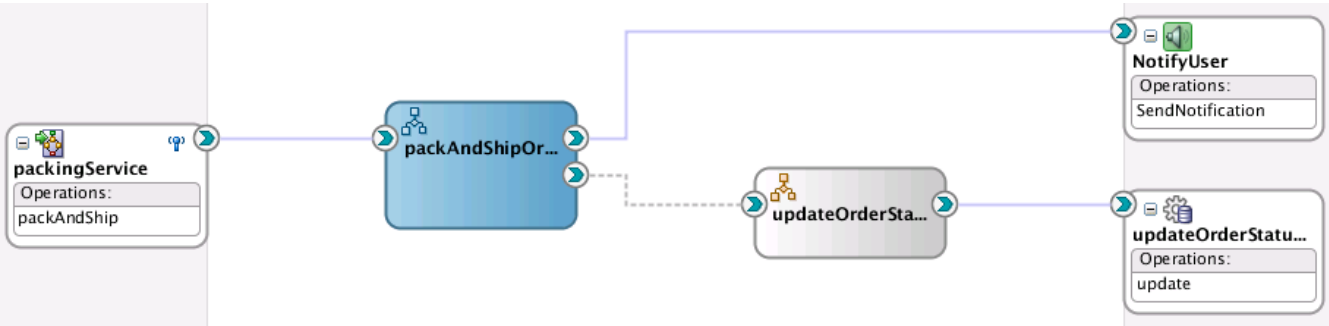
Component	How This Component Addresses the Business Challenge	Component Description
REST service	An inbound REST adapter service defines a shipping resource.	REST is an architecture for designing network applications. RESTful applications use HTTP requests to post data (create and update), get data (for example, make queries), update data, and delete data.
BPEL process	A BPEL process sets the status of the order to shipped, notifies the customer that the order has shipped, and updates the order status in the database. This process is connected to the REST interface, which is exposed as a service.	BPEL processes provide process orchestration and storage of a synchronous or an asynchronous process. You design a business process that integrates a series of business activities and services into an end-to-end process flow.
HTTP Analyzer	The HTTP Analyzer tests the Web Application Description Language (WADL) URL of the REST service to ensure that the service is working properly.	The HTTP Analyzer enables you to examine the content of HTTP request/response package pairs. You can edit the content of a request package, resend it, and observe the response packet returned.

Table 6-1 (Cont.) Components That Provide the Business Solution

Component	How This Component Addresses the Business Challenge	Component Description
BPEL process component template	The component template includes an invoke activity and an assign activity that takes an order number and an order status and assigns it to the input variable of the database adapter.	See Table 3-1 for a description of Oracle SOA Suite templates.
Standalone BPEL subprocess	A standalone BPEL process calls a database adapter reference to update the order status.	A standalone BPEL subprocess is a fragment of a BPEL process, which includes a number of activities to reuse. Standalone subprocess do not have an interface and are only called from another BPEL process.
Composite sensor	A composite sensor tracks all orders of a specific shipping provider.	See Table 3-1 for a description of composite sensors.
Oracle Enterprise Manager Fusion Middleware Control	Oracle Enterprise Manager Fusion Middleware Control monitors the composite sensor in the instance and provides access to a WADL URL used for testing.	Oracle Enterprise Manager Fusion Middleware Control is a web browser-based, graphical user interface that you use to monitor and administer your deployed composites.
Oracle User Messaging Service (UMS) adapter	A UMS adapter sends an email notification to a customer indicating the order was shipped.	UMS is an Oracle Fusion Middleware component that enables communication between users and applications. UMS supports various messaging channels such as email, SMS, and instant messaging (IM). UMS consists of servers, drivers, and applications.

Figure 6-1 provides an overview of how this business solution is implemented.

Figure 6-1 Packing and Shipping Process Overview



Subsequent sections of this chapter provide more specific details about how the components in Table 6-1 are used to address the packing and shipping business challenge.

- Defining a Shipping Resource with a REST Service
- Exposing a REST Service with a Packing BPEL Process
- Testing REST Services with the HTTP Analyzer
- Using Templates and Standalone Subprocesses to Update the Order Status in the Database

- Tracking the Shipping Provider with Composite Sensors
- Sending Email Notifications to Indicate Order Shipments

Defining a Shipping Resource with a REST Service

Company X creates a SOA composite application named **PackAndShipService** to address this business challenge. As an alternative to using a web service, Company X uses a REST service in the composite to define a shipping resource.

REST is an architecture for designing network applications. RESTful applications use HTTP requests to post data (create and update), get data (for example, make queries), update data, and delete data.

Oracle SOA Suite provides the following REST support:

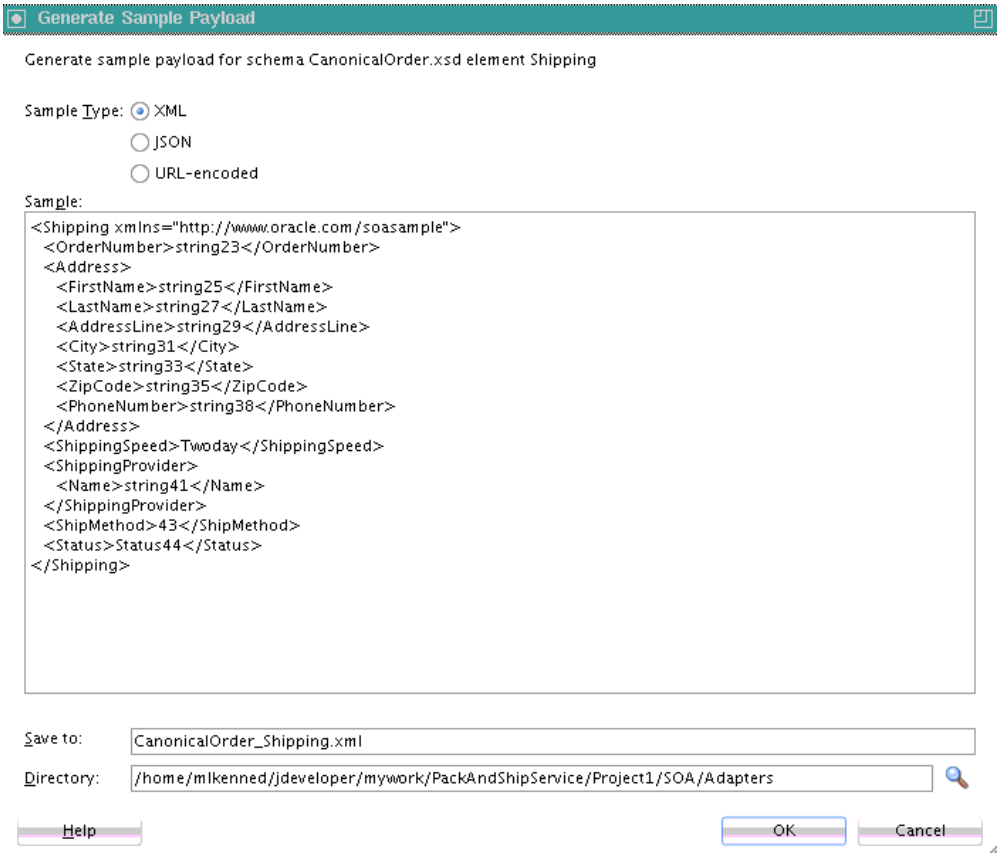
- Support in SOA composite applications:
  - Enable REST support in new or existing services.
  - Integrate with external REST APIs.
  - Orchestrate a set of RESTful state transitions (RPC/Hypermedia as the Engine of Application State (HATEOAS) approach).
  - Support for XML, JavaScript Object Notation (JSON) (with automatic translation to and from XML), URI sample, and URL-encoded GET/POST data.
  - Generation of sample URI for REST service operations.
  - Support for WADL services. The WADL can be provided by a deployed Oracle SOA Suite or Oracle Service Bus service or a non-Oracle SOA Suite or Oracle Service Bus service such as a Jersey REST service.
- Ease of development:
  - Oracle JDeveloper provides a wizard for modeling REST interfaces and WSDL operation bindings.
  - Readable API is provided that publishes each method used upon deployment.
  - Ability to browse and consume Oracle REST endpoints (including Oracle Service Bus) from within Oracle JDeveloper.
- Oracle Web Service Manager (OWSM) policy support for REST security.

Company X drags a REST binding adapter from the Components window into the **Exposed Services** swimlane in Oracle JDeveloper. The packing service expects a shipping resource that includes all necessary information to pack and ship an order. It returns a shipping resource with an updated order status.

Company X defines the following REST service details:

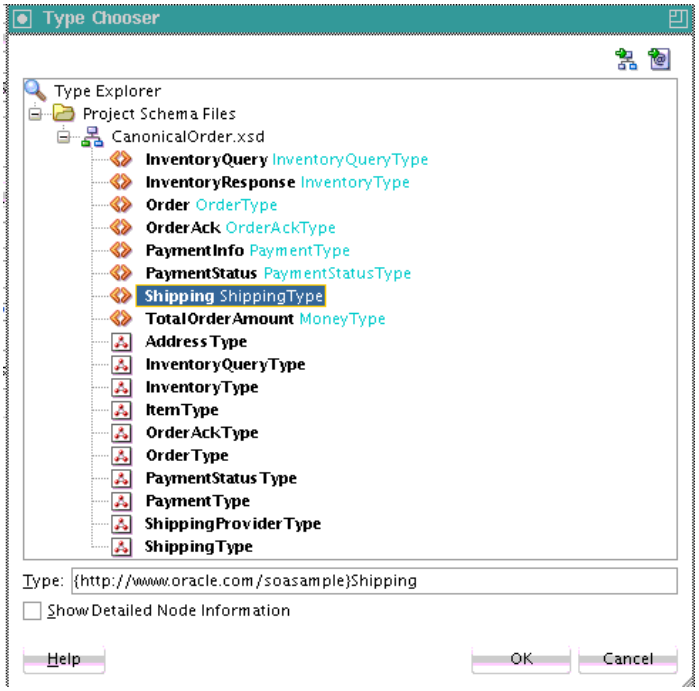
- Creates a REST inbound interface.
- Creates a **shipping** REST resource.
- Creates a **packandShip** operation binding with:
  - A **POST** HTTP verb.
  - A sample request XML payload, as shown in Figure 6-2.

Figure 6-2 Sample XML Payload



– A request schema file selected from the **ProcessOrder** project created in Creating a SOA Composite Application From a SOA Project Template. The schema file includes the **Shipping** element, as shown in Figure 6-3.

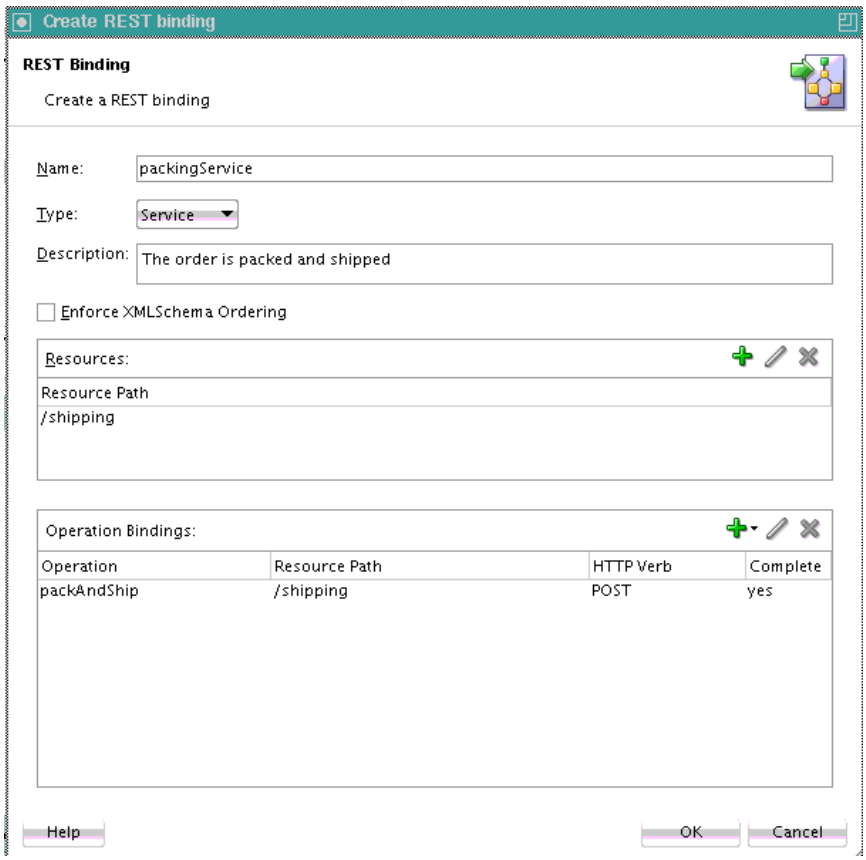
Figure 6-3 Shipping Element in Schema File



– A response schema file selected from the **ProcessOrder** project. The schema file is the same as selected for the request and also includes the **Shipping** element shown in Figure 6-3.

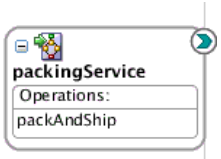
When design is complete, the Create REST Binding dialog looks as shown in Figure 6-4.

Figure 6-4 Create REST Binding Dialog - Design Complete



This creates a REST service binding component in the **Exposed Services** swimlane, as shown in Figure 6-5.

Figure 6-5 REST Service in Exposed Services Swimlane in Oracle JDeveloper



Exposing a REST Service with a Packing BPEL Process

Company X exposes the REST service by connecting it to a BPEL process, as shown in Figure 6-6.

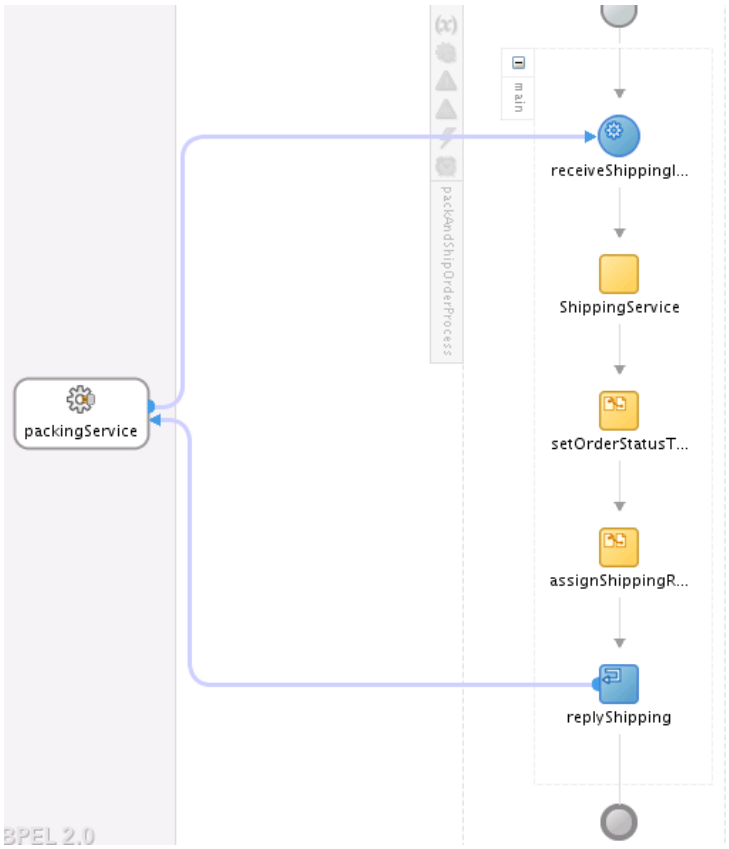
Figure 6-6 REST Service Connected to BPEL Process

This service initiates the packing and shipping based on shipping provider and shipping method, which are both determined by the order fulfillment service in Fulfilling Orders . The BPEL process consists of the following activities:

- A receive activity receives a call from the REST service.
- An empty activity simulates the packing and shipping. After this activity is processed, the order status is updated to **Shipped**.
- An assign activity assigns the order number and status to two variables:
  - **orderNumber** is set from the input variable
  - **orderStatus** is set to **Shipped**
- An assign activity assigns the value of the input variable (with updated status) to the output variable.
- A reply activity returns information to the REST service.

The complete design of the BPEL process is shown in Figure 6-7.

Figure 6-7 REST Service Integration with the BPEL Process



Testing REST Services with the HTTP Analyzer

To ensure that the REST service is working properly, Company X first deploys the SOA composite application for testing with the HTTP Analyzer.

Deployment is performed by right-clicking the **PackAndShipService** in the Applications window, selecting **Deploy**, and going through the pages of the Deploy wizard. Deployment is

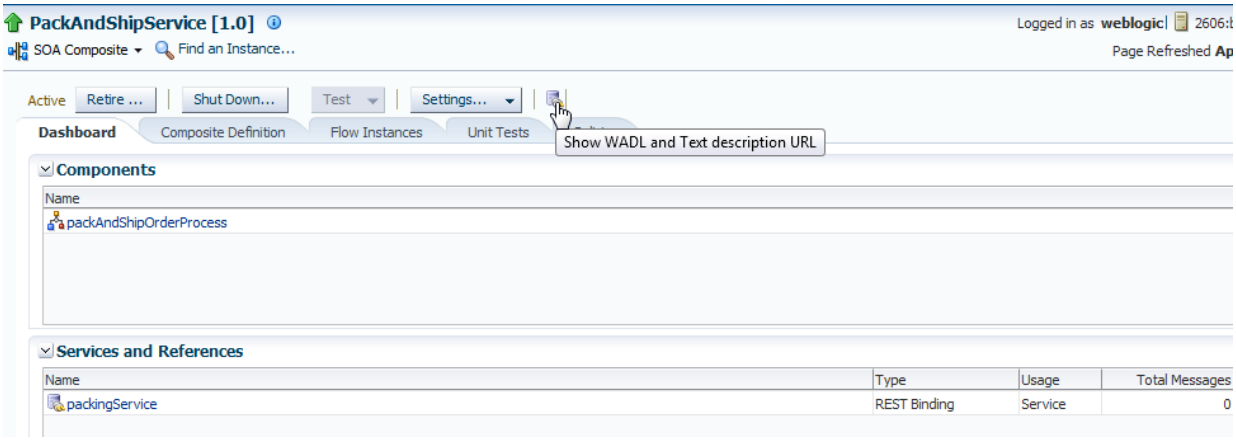
performed to the **IntegratedWebLogicServer**, an embedded, local server in Oracle JDeveloper.

The HTTP Analyzer enables you to examine the content of HTTP request/response package pairs. You can edit the content of a request package, resend it, and observe the response packet returned.

To test the service, Company X first copies the WADL file URL from the home page of the **PackAndShipService** SOA composite application in Oracle Enterprise Manager Fusion Middleware Control, as shown in Figure 6-8.

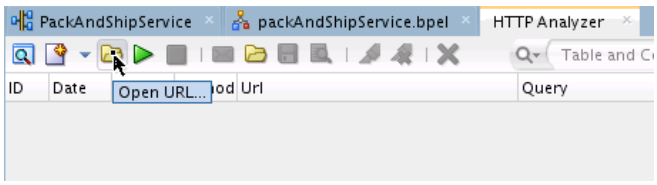
REST services in Oracle SOA Suite expose WADL files instead of WSDL files to define their interface. WADL provides a readable XML description of HTTP-based web applications (typically REST web services). WADL simplifies the reuse of web services based on the existing HTTP architecture of the web.

Figure 6-8 WADL URL Location in Oracle Enterprise Manager Fusion Middleware Control



Company X then opens the HTTP Analyzer in Oracle JDeveloper by selecting **Tools** > **HTTP Analyzer**. Company X clicks **Open URL** and enters the WADL URL copied from Oracle Enterprise Manager Fusion Middleware Control, when prompted. Figure 6-9 provides details.

Figure 6-9 HTTP Analyzer

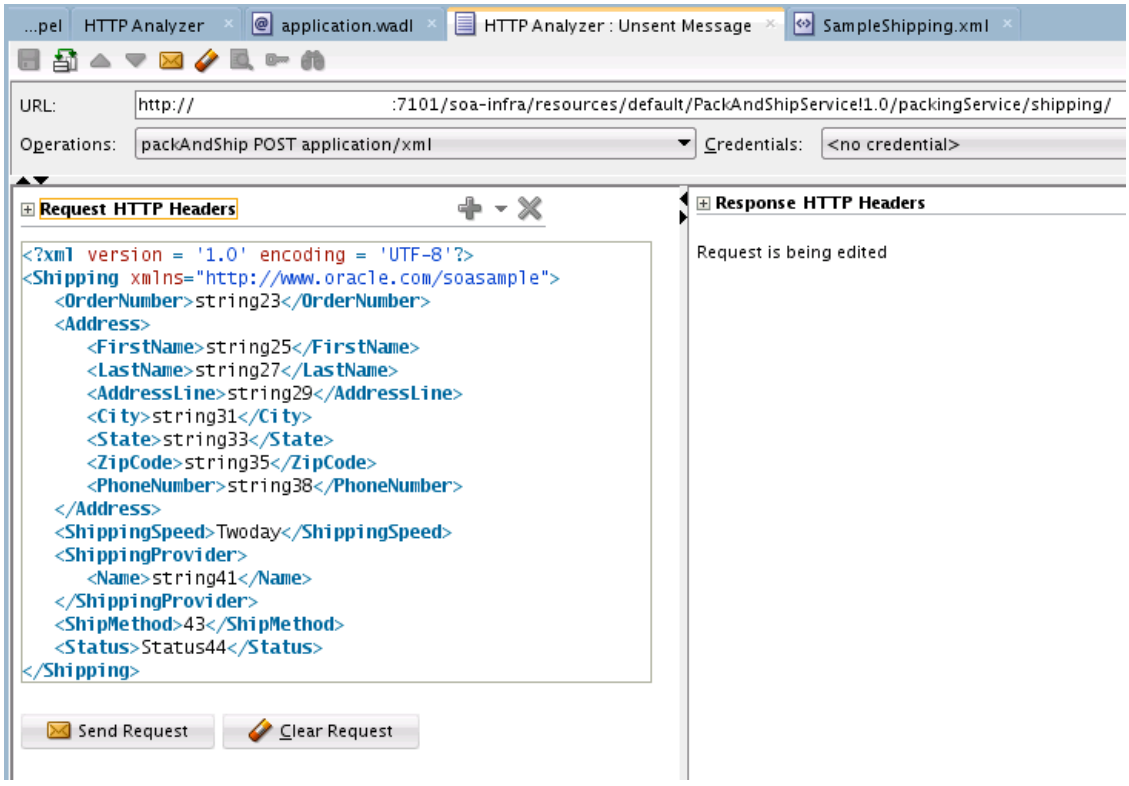


The Service dialog shown in Figure 6-10 displays details about the REST service. Company X clicks **Test**.

Figure 6-10 Service Dialog

Company X copies and pastes the sample request XML payload specified in Figure 6-2 into the **Request HTTP Headers** section and clicks **Send Request**, as shown in Figure 6-11.

Figure 6-11 Request XML Payload Input to HTTP Analyzer

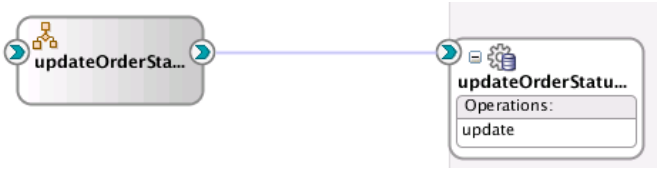


Using Templates and Standalone Subprocesses to Update the Order Status in the Database

In addition to updating the order status in the shipping message, Company X also updates it in the database. Because this is a common task, Company X creates a component template named **updateOrderStatusSP** for reuse as needed in multiple projects. Company X used templates for similar common tasks in Calculating Payment Status with XSLT Transformations and Customizing the Contents of the SOA Project Template.

Figure 6-12 shows the component template contents.

Figure 6-12 BPEL Process Component Template Contents



Company X drags the template from the **Component Templates** section of the Components window into the SOA composite application. When fully expanded, the composite looks as shown in Figure 6-13.

Figure 6-13 SOA Composite Application with Expanded Template

The component template includes an invoke activity and an assign activity that takes an order number and an order status and assigns it to the input variable of the database adapter.

The **packAndShipOrder** BPEL process must be connected with the **updateOrderStatusSP** component template. To address this task, Company X uses a standalone BPEL subprocess to call a database adapter reference to update the order status. Subprocesses are similar to templates in that they enable you to reuse functionality in multiple projects. A standalone BPEL subprocess is a fragment of a BPEL process, which includes a number of activities to reuse. Standalone subprocesses do not have an interface and are only called from another BPEL process. A standalone process can have partner links across a number of other BPEL processes.

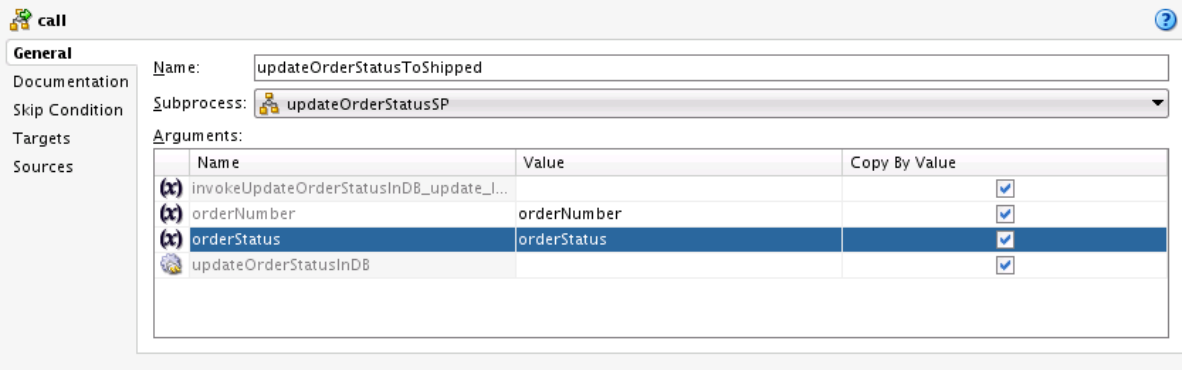
Company X opens the **packAndShipOrder** BPEL process and drags a standalone BPEL process from the **Standalone** section of the Components window into the process.

The standalone BPEL process is displayed as a call activity, which calls the **updateOrderStatusSP** template. It provides the following:

- Order number
- Order status

Figure 6-14 shows the contents of the call activity. A call activity enables you to execute referenced subprocess code in standalone and inline subprocesses.

Figure 6-14 Contents of Call Activity



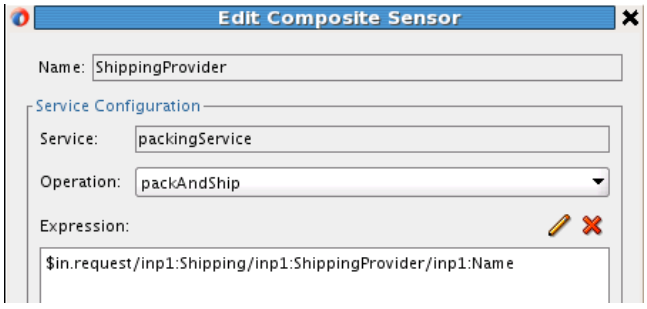
Tracking the Shipping Provider with Composite Sensors

Company X added a composite sensor for tracking the status of order payments in Tracking Payment Status with Composite Sensors and tracking the order number in Tracking the Order Number with Composite Sensors.

Company X now has an additional requirement for a composite sensor to track the shipping provider. This enables Company X to search for all orders that have been shipped with a specific shipping provider.

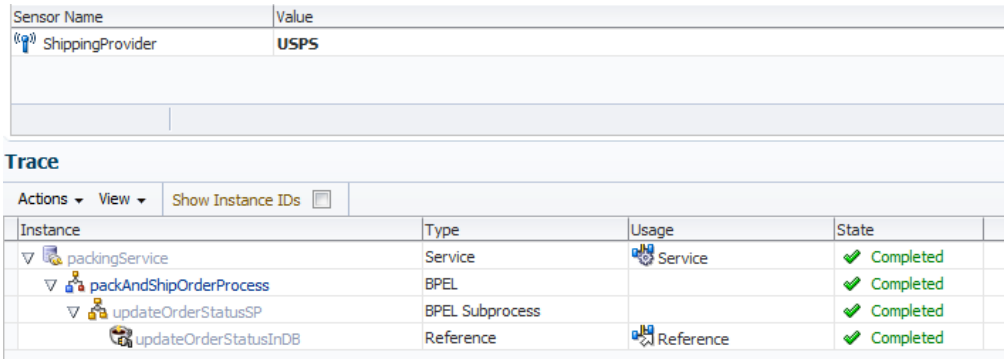
Company X defines the composite sensor on the REST service. The definition includes an XPath expression to track the shipping provider, as shown in the Composite Sensor dialog in Figure 6-15.

Figure 6-15 Composite Sensor for Tracking the Shipping Order



The **Enterprise Manager** check box of the Composite Sensor dialog is also selected. This enables you to track composite sensor names and values on the Flow Instances page or the Flow Trace page for a specific business flow instance in Oracle Enterprise Manager Fusion Middleware Control. Figure 6-16 provides details.

Figure 6-16 Composite Sensor Name and Value in Flow Trace Page



Sending Email Notifications to Indicate Order Shipments

Company X must be able to notify customers by email that an order has shipped. To address this task, Company X configures a UMS adapter to send an email notification to customers.

UMS is an Oracle Fusion Middleware component that enables communication between users and applications. UMS supports various messaging channels such as email, SMS, and instant messaging (IM). UMS consists of the following components:

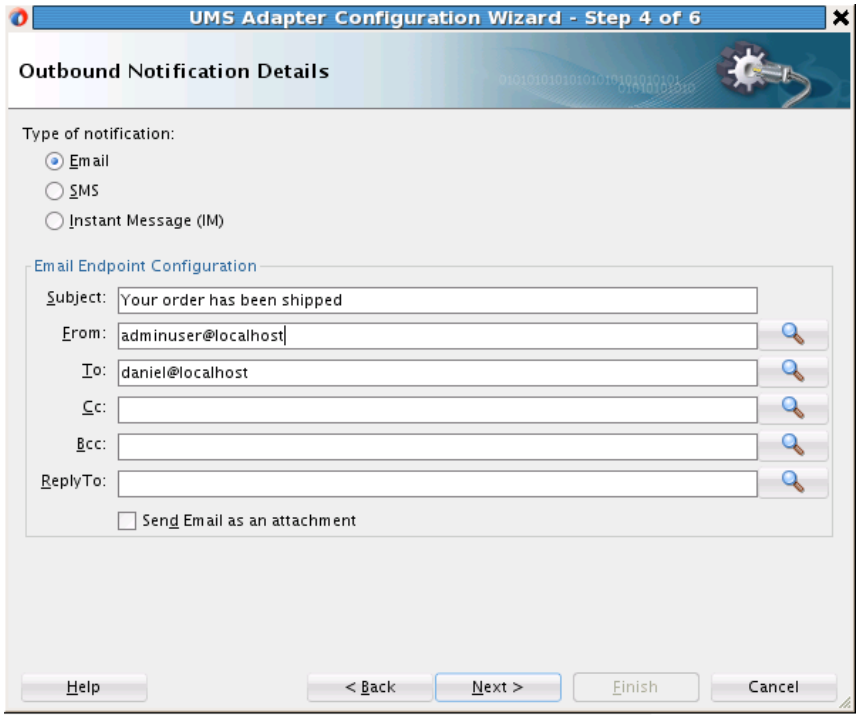
- A UMS server that orchestrates message flows between applications and users.
- UMS drivers that connect UMS to the messaging gateways, adapting content to the various protocols supported by UMS.
- UMS client applications that implement the business logic of sending and receiving messages.

The UMS adapter is dragged from the **Technology** section of the Components window to the **External References** swimlane. Company X configures the UMS adapter to perform the following tasks:

- Send an outbound notification
- Uses email as the notification channel
- Add an email subject of `Your Order Has Been Shipped`
- Configure the **From** and **To** email addresses

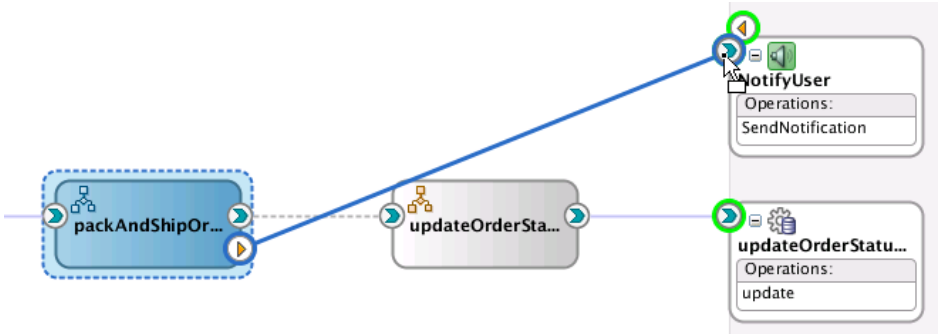
Figure 6-17 provides details.

Figure 6-17 UMS Adapter Configuration Wizard



When UMS configuration is complete, the **packAndShipOrder** BPEL process is connected to the **NotifyUser** UMS adapter reference. Figure 6-18 provides details.

Figure 6-18 BPEL Process Connected to the UMS Adapter

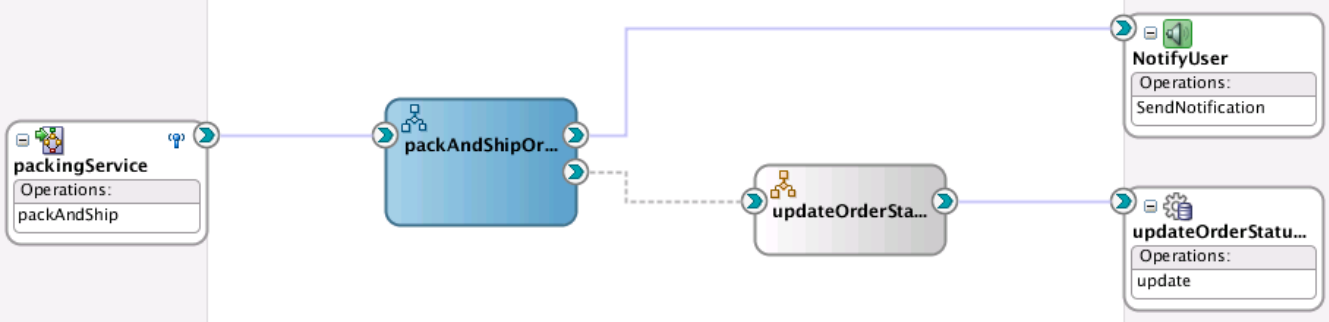


To complete configuration, Company X adds the appropriate activities to the **packAndShipOrder** BPEL process:

- An invoke activity to invoke the UMS adapter (with an input variable)
- An assign activity to populate the payload of the input variable of the invoke activity

Figure 6-19 provides an overview of how this completed business solution appears in the SOA Composite Editor.

Figure 6-19 Completed SOA Composite Application



Related Documentation

Table 6-2 provides references to documentation that more specifically describes the components and features described in this chapter.

Table 6-2 Related Documentation

For Information About...	See...
Integrating a REST operation	Integrating REST Operations in SOA Composite Applications in *Developing SOA Applications with Oracle SOA Suite*
Creating and designing a BPEL process	Getting Started with Oracle BPEL Process Manager in *Developing SOA Applications with Oracle SOA Suite*
Using the HTTP Analyzer	Monitoring HTTP Using the HTTP Analyzer in *Developing Applications with Oracle JDeveloper*
Creating Oracle SOA Suite templates and standalone BPEL subprocesses	Oracle SOA Suite Templates and Reusable Subprocesses in *Developing SOA Applications with Oracle SOA Suite*
Creating composite sensors	Defining Composite Sensors in *Developing SOA Applications with Oracle SOA Suite*
Configuring the UMS Adapter	Oracle JCA Adapter for UMS in *Understanding Technology Adapters*
Using the Integrated WebLogic Server	Introducing the Quick Start Distributions in *Installing SOA Suite and Business Process Management Suite Quick Start for Developers*

7

Fulfilling Orders

This chapter describes how Oracle SOA Suite addresses the business challenge of creating an order fulfillment system. Overviews of how key SOA composite application components are used to address this challenge are provided, including project templates, business rules, composite sensors, REST binding references, and Coherence adapters.
This chapter includes the following sections:

- Business Challenge
- Business Solution
- Related Documentation

Business Challenge

Company X has a requirement to create a system that listens for orders to be processed, selects a shipping provider (for example, the United States Postal Service (USPS) or UPS), and invokes a packing and shipping service. In this scenario, the packing and shipping service to invoke is the **PackAndShipService** composite that was designed in Packing and Shipping Orders .

Business Solution

To address this business challenge, Company X designs a business solution that uses the components described in Table 7-1.

Table 7-1 Components That Provide the Business Solution

Component	How This Component Addresses The Business Challenge	Component Description
SOA composite application that includes a project template	An order fulfillment composite created from a project template listens for orders to be processed, selects a shipping provider, and invokes the packing and shipping service (**PackAndShipService**) created in Packing and Shipping Orders . The order fulfillment composite is triggered when an order is updated as **ReadyForShip** in the database. It then locates the shipping speed in the order message, determines the shipping method based on the shipping speed and shipping state, reads the preferred shipping provider from the database or Coherence cache, and calls the packing and shipping REST service.	See Table 3-1 for a description of templates.

Table 7-1 (Cont.) Components That Provide the Business Solution

Component	How This Component Addresses The Business Challenge	Component Description
Business rule	A business rule (decision table) decides the shipping method based on speed and shipping state. Based on the shipping method, the preferred shipping provider is retrieved from the database.	Business rules enable dynamic decisions at runtime that allow you to automate policies, computations, and reasoning while separating rule logic from underlying application code. This enables more agile rule maintenance and empowers business analysts with the ability to modify rule logic without programmer assistance and without interrupting business processes.
Composite sensor	A composite sensor tracks the order number.	See Table 3-1 for a description of composite sensors.
REST reference	The outbound REST reference delivers the order to the packing and shipping service.	See Table 6-1 for a description of REST bindings.
Coherence adapter	A Coherence adapter initially reads the correct shipping provider from the database, and from Coherence cache for subsequent read operations. An additional Coherence adapter copies the database adapter response into Coherence cache so that the shipping provider is available in cache the next time it is looked up.	A Coherence cache is a collection of data objects that serves as an intermediary between the database and the client applications. Database data can be loaded into a cache and made available to different applications. Coherence cache reduces load on the database and provides faster access to database data.

Subsequent sections of this chapter provide more specific details about how the components in Table 7-1 are used to address the order fulfillment business challenge.

- Creating a Project from a SOA Template
- Determining the Shipping Method with a Business Rule
- Tracking the Order Number with Composite Sensors
- Delivering the Order to the Packing Service with the REST Interface
- Reading the Shipping Provider from Cache with the Coherence Adapter
- Copying the Database Adapter Response into Coherence Cache
- Deploying the Composite and Testing the Coherence Adapters

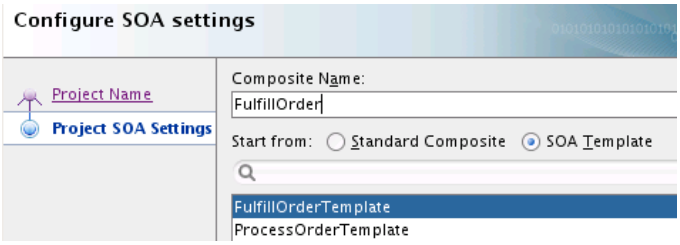
Creating a Project from a SOA Template

Company X frequently has business requirements for designing SOA composite applications that listen for orders to be processed, select a shipping provider, and invoke a packing and shipping service. To address this challenge, Company X created a project template named **FulfillOrderTemplate** with these capabilities that can be imported into multiple applications in Oracle JDeveloper, as necessary. The template can then be customized for the business requirements of that specific project. Changes made to that specific imported template are not propagated to projects previously created using this template.

As with previous templates, the **FulfillOrderTemplate** project template is first registered in Oracle JDeveloper by selecting **Tools** > **Preferences** > **SOA** > **Templates**, and specifying the template storage location.

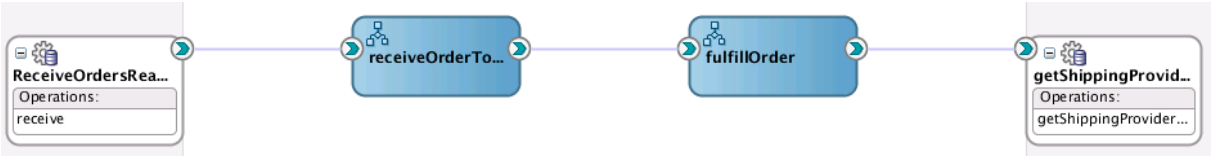
Company X invokes the Create SOA Project wizard to create a new SOA project, and selects to create one based on a template. The project template is imported into the new application by selecting **SOA Template** in the Create SOA Project wizard, which refreshes the dialog to display existing templates for selection. **FulfillOrderTemplate** is selected, then the project name is shortened to **FulfillOrder**, as shown in Figure 7-1.

Figure 7-1 Project Template Selection



When creation is complete, the composite looks as shown in Figure 7-2 with the imported project template.

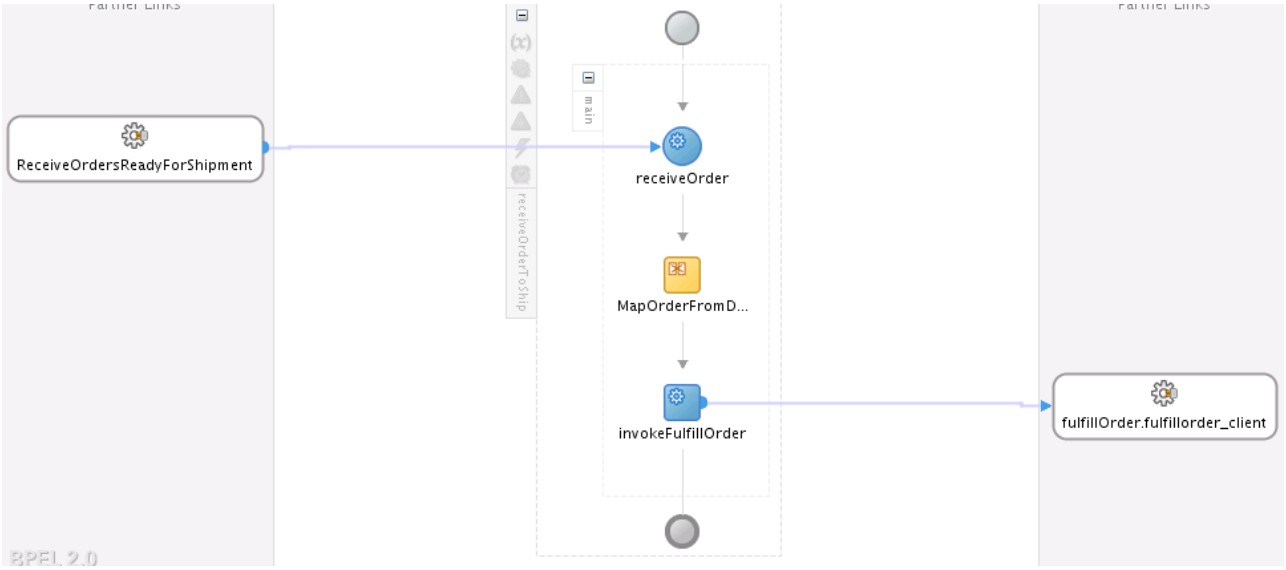
Figure 7-2 Contents of the SOA Composite Application



The imported project template consists of the following components:

- The first database adapter (**ReceiveOrdersReadyForShipment**) listens for orders with a status of **ReadyForShip**, reads the record out of the database, and triggers a new BPEL process for each order. To prevent the order from being read again, it changes the status to **ReadyForPack**.
- The second database adapter (**getShippingProvider**) reads the shipping provider from the database, using the shipping method ID as a primary key. A list of preferred shipping providers per shipping method is maintained in the database. For example, USPS is used for USAFirstClass shipment (shipping method ID =1), UPS is used for USAPriority shipment (shipping method ID =2), and so on.
- The **receiveOrderToShip** BPEL process is open and displayed in Figure 7-3. This process is invoked by the **ReceiveOrdersReadyForShipment** database adapter service. An XSLT map transforms the message into a canonical order message. The **fulfillOrder** BPEL process is invoked with the canonical order message as input.

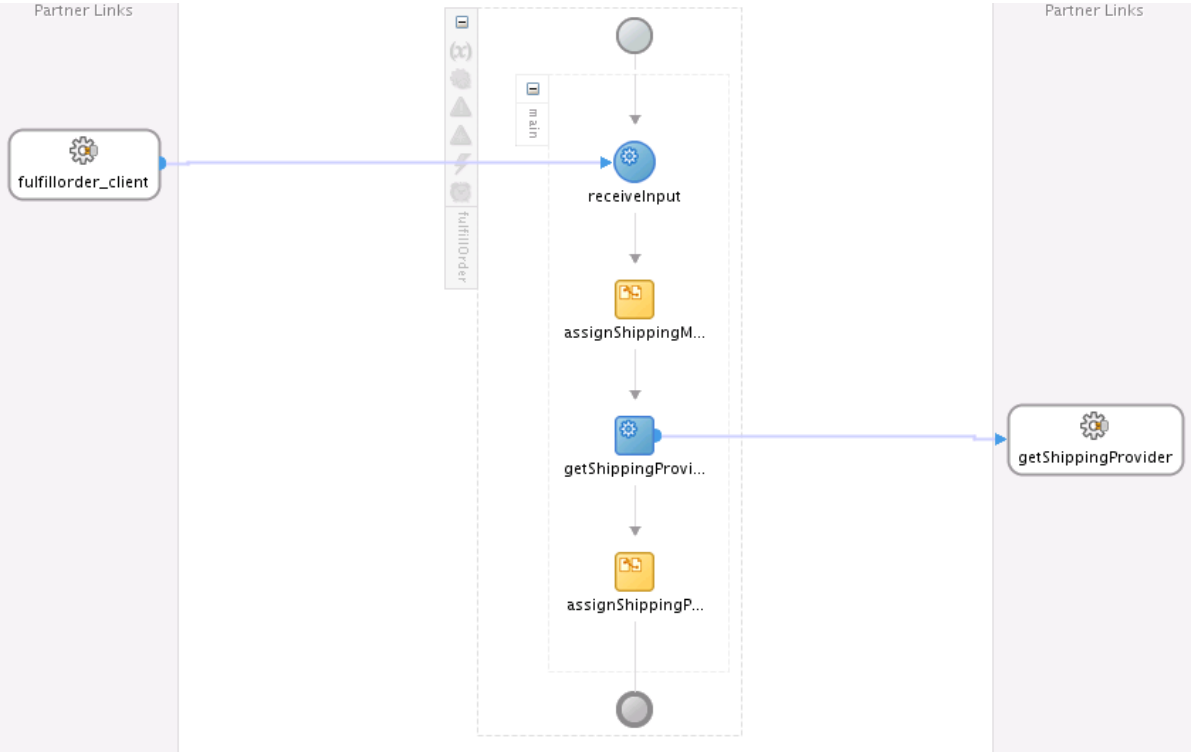
Figure 7-3 receiveOrderToShip BPEL Process Contents



- The **fulfillment** BPEL process is shown in Figure 7-4. The incoming order message includes the shipping speed selected by the customer:
  - One-day shipping
  - Two-day shipping
  - Standard shipping: 3-5 business days
  - Shipping speed does not matter

  A business rule determines the shipping method based on shipping speed and shipping address. The shipping method ID is used as input to the database call to retrieve the shipping provider.

Figure 7-4 fulfillment BPEL Process Contents



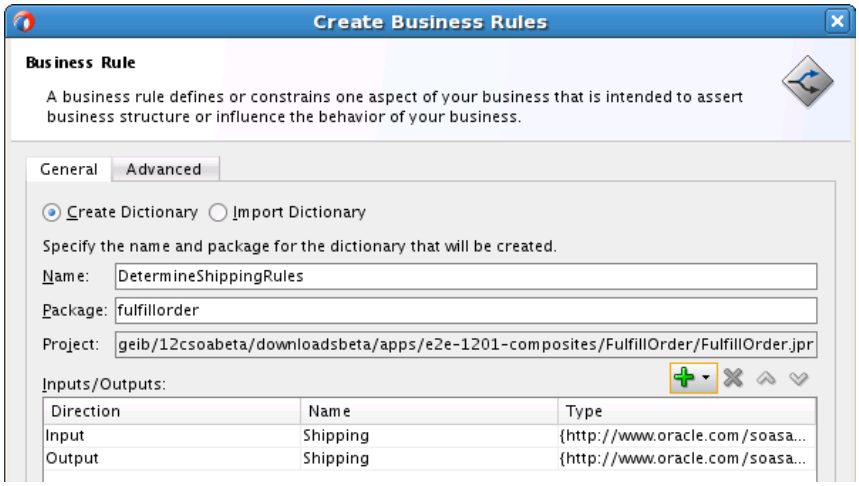
Determining the Shipping Method with a Business Rule

Business rules enable dynamic decisions to be made at runtime. This enables you to automate policies, computations, and reasoning while separating rule logic from underlying application code.

Company X adds a business rule to determine the shipping method ID. The rule takes as input the shipping state and the requested shipping speed and returns the shipping method ID. For example, USPS is used for USAFirstClass shipment (shipping method ID =1), UPS is used for USAPriority shipment (shipping method ID =2), and so on.

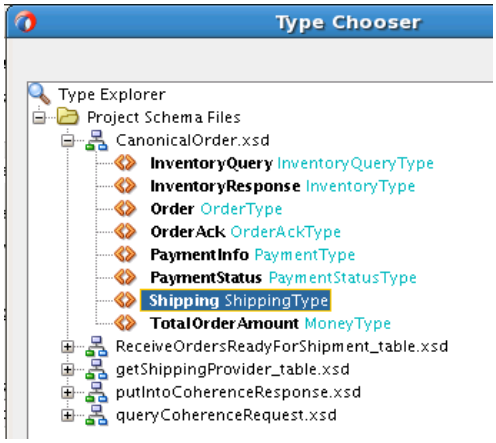
Company X drags a **Business Rule** icon from the **Components** section of the Components window into the SOA composite application. After configuration is complete, the Create Business Rules dialog looks as shown in Figure 7-5.

Figure 7-5 Input and Output Facts of Defined Business Rule



Both the input and output facts shown in Figure 7-5 are defined with the **Shipping** element type in the schema file, as shown in Figure 7-6. Facts are the objects on which rules reason. Each fact is an instance of a fact type.

Figure 7-6 Shipping Element Type



After business rule creation, the business rule service component in the composite is clicked to access the Rules Editor. Within the Rules Editor, a decision table is defined, as shown in Figure 7-7. A decision table is an alternative business rule format that is more compact and intuitive when many rules are needed to analyze many combinations of property values. You can use a decision table to create a set of rules that covers all combinations or where no two combinations conflict.

The following rule conditions are defined for speed of delivery. A rule condition represents the IF part of a statement. A rule condition is like a query over the available facts in the Rules Engine, and for every row returned from the query, the rule is activated. The rule condition activates the rule whenever a combination of facts makes the conditional expression true.

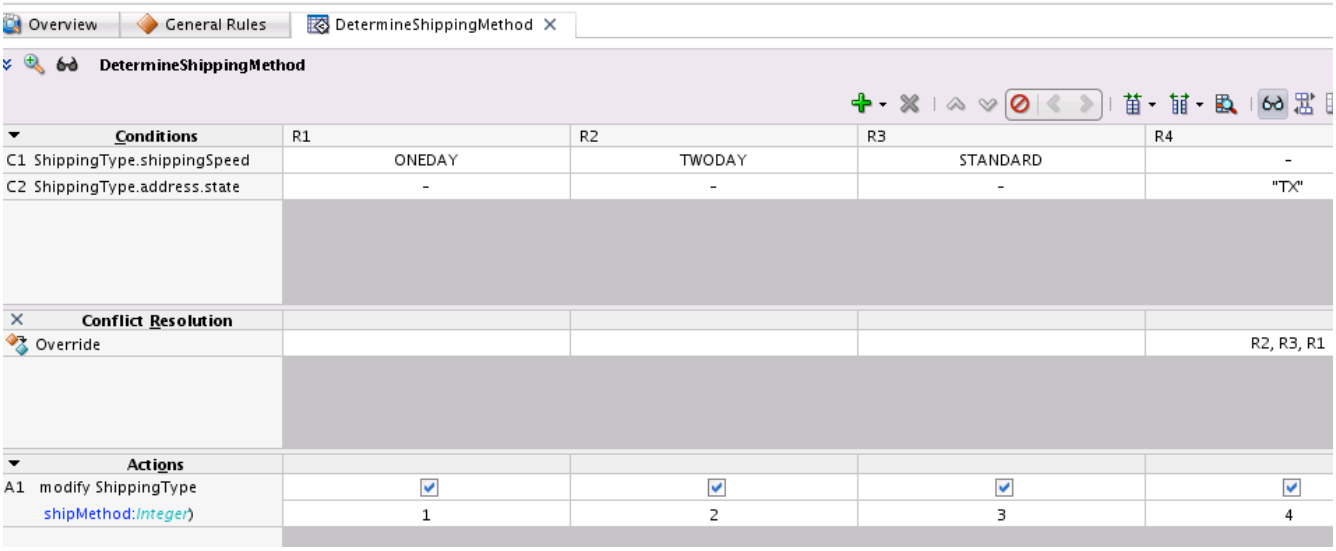
- One-day shipping
- Two-day shipping

- Standard shipping: 3-5 business days
- Do not care (shipping speed does not matter). For this delivery method, a state condition of TX (for Texas) is set. No other rule conditions have a state set; delivery speed is the only requirement.

The following shipping method ID rule actions are defined for each rule condition. A rule action represents the THEN part of a statement. The THEN part contains the actions that are executed when the rule is fired. A rule is fired after it is activated and selected among the other rule activations using conflict resolution mechanisms such as priority.

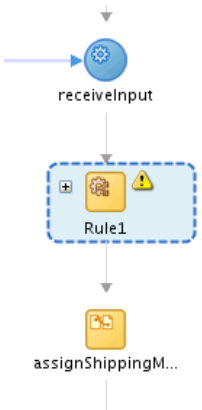
- 1 - USPS is used for USAFirstClass shipping
- 2 - UPS for USAPriority shipping
- 3 - USA free mail
- 4 - UPS next day air

Figure 7-7 Decision Table in Rules Editor



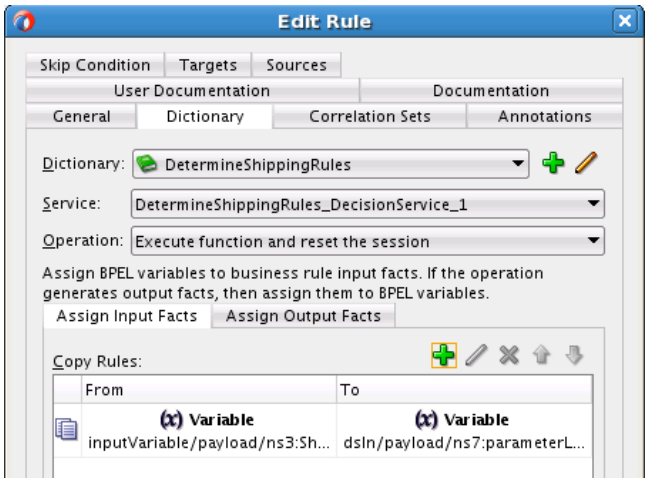
The business rule is integrated by dragging a **Business Rule** icon from the **SOA Components** section of the Components window into the **fulfillment** process, as shown in Figure 7-8.

Figure 7-8 Business Rule Integrated into fulfillment BPEL Process



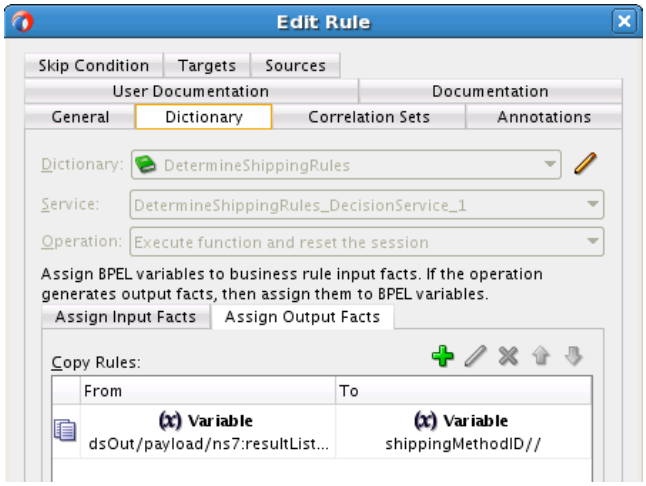
Company X selects the dictionary invoked by this activity. The input and output facts are also defined. For the input copy rule, **ShippingType** is copied from the process **inputVariable** to the **dsIn** (input fact) variable of the business rule. Figure 7-9 provides details.

Figure 7-9 Input Copy Rule



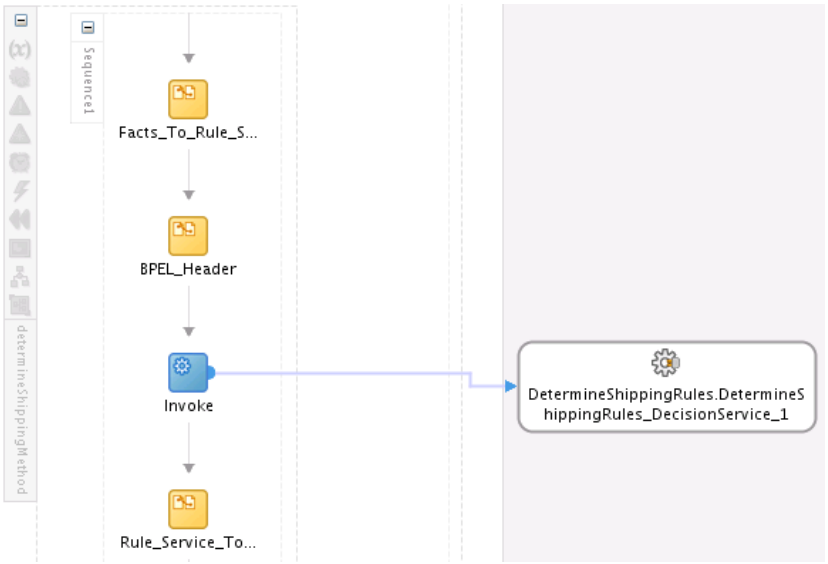
For the output copy rule, **ShippingMethod** is copied from the **dsOut** (output fact) variable of the business rule to the **shippingMethodID** process variable. Figure 7-10 provides details.

Figure 7-10 Output Copy Rule



When design is complete, the **fulFillment** process looks as shown in Figure 7-11.

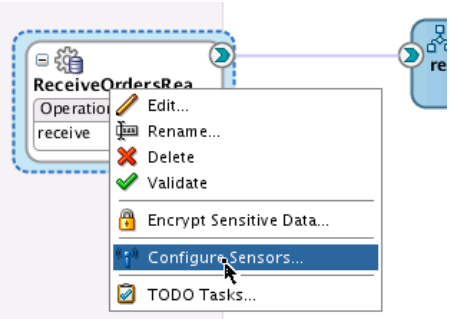
Figure 7-11 Invocation of Rules Dictionary



Tracking the Order Number with Composite Sensors

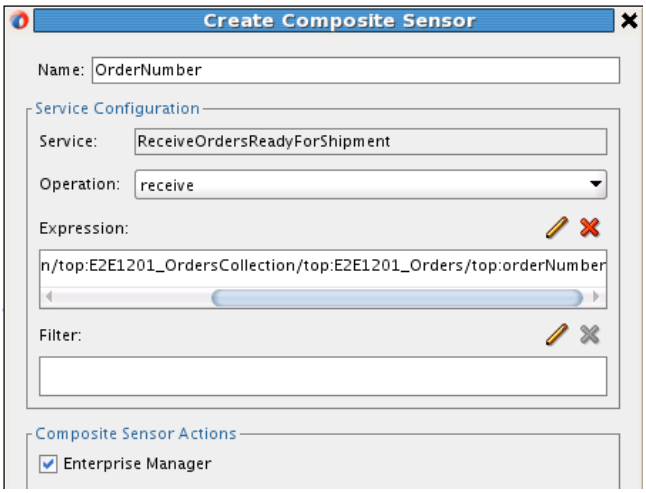
As performed in previous chapters, Company X adds a composite sensor for tracking fields on messages. For this scenario, the status of the order number is tracked. The sensor is set on the **ReceiveOrderReadyForShipment** database adapter service. Figure 7-12 provides details.

Figure 7-12 Composite Sensor



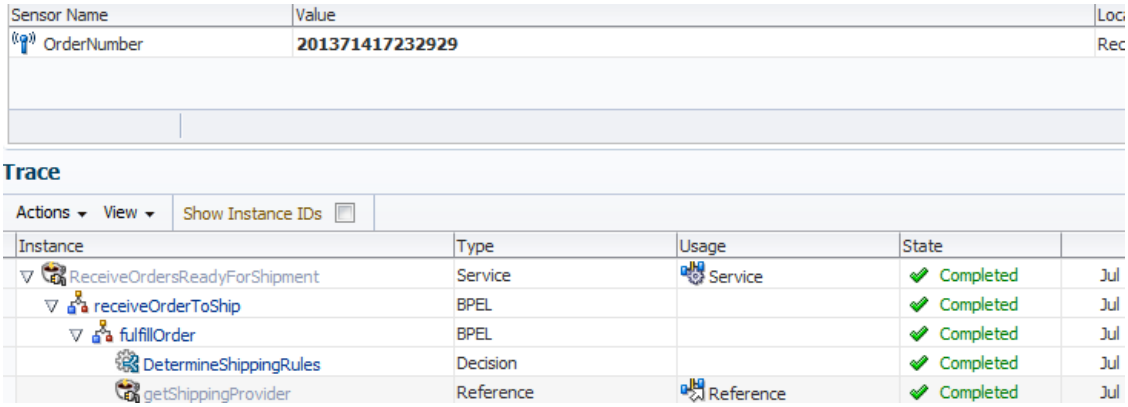
The definition includes an XPath expression to track the order number, as shown in the Composite Sensor dialog in Figure 7-13.

Figure 7-13 Composite Sensor Dialog



The **Enterprise Manager** check box of the Create Composite Sensor dialog is also selected. This enables you to track composite sensor names and values on the Flow Instances page or the Flow Trace page for a specific business flow instance in Oracle Enterprise Manager Fusion Middleware Control. Figure 7-14 provides details.

Figure 7-14 Composite Sensor Name and Value on Flow Trace Page



Delivering the Order to the Packing Service with the REST Interface

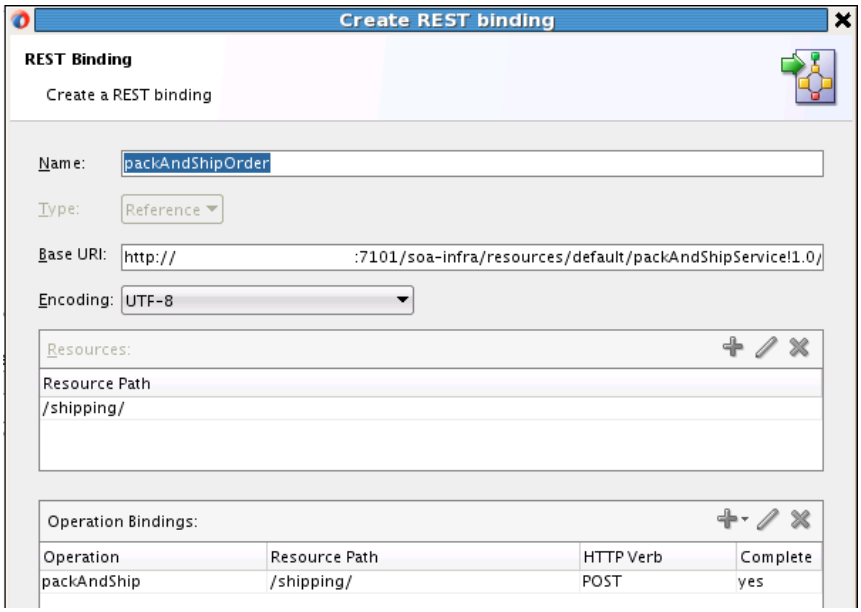
A REST service was used as an alternative to web services in Exposing a REST Service with a Packing BPEL Process. REST uses HTTP requests to post data (create and update), get data (for example, make queries), update data, and delete data.

For this business scenario, Company X uses a REST reference to call the **PackAndShip** service created in Packing and Shipping Orders . Company X drags a REST binding into the **External References** swimlane of the composite and defines the following details:

- A REST outbound interface.
- A **shipping** REST resource.
- A **packandShip** operation binding based on a WADL file with a **POST** HTTP verb. WADL provides a readable XML description of HTTP-based web applications (typically REST web

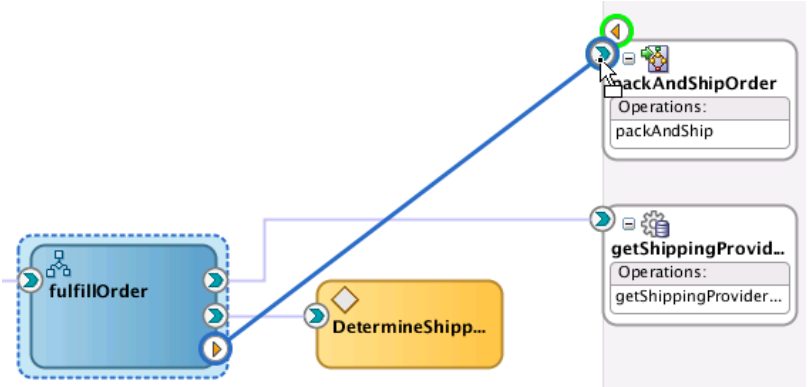
services). WADL simplifies the reuse of web services based on the existing HTTP architecture of the web. When you select a WADL file, all operation binding details are automatically populated in the Create REST Binding dialog. Figure 7-15 provides details.

Figure 7-15 REST Binding



When configuration is complete, Company X connects the **fulfillOrder** BPEL process to the REST reference, as shown in Figure 7-16.

Figure 7-16 BPEL Process Connected to REST Reference



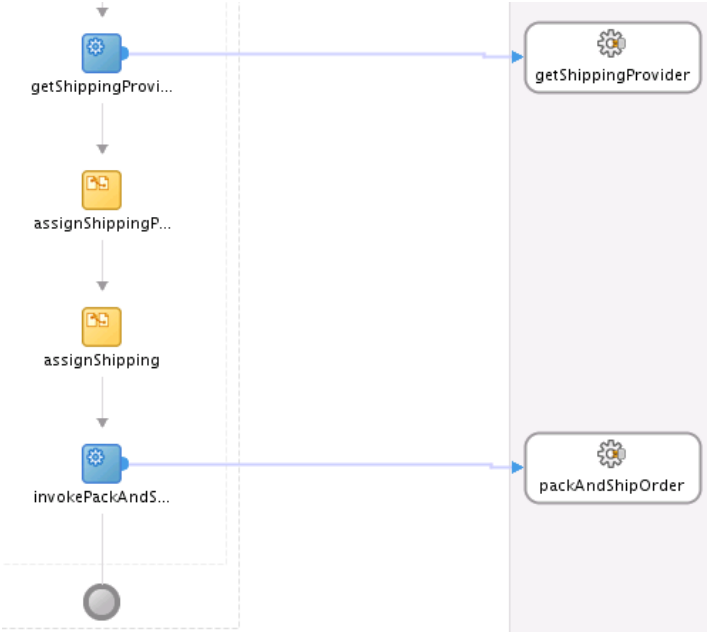
To complete configuration, Company X adds the necessary activities to the **fulfillOrder** process:

- An invoke activity invokes the **PackAndShipOrder** partner link and defines the necessary input and output variables.
- An assign activity maps the **shipping** element in the input variable to the **shipping** element in the input variable for the REST reference.

Because the BPEL process is one-way, there is no need to assign the return value of the REST service.

Figure 7-17 provides details.

Figure 7-17 fulFillment BPEL Process Contents



Reading the Shipping Provider from Cache with the Coherence Adapter

The database adapters used in this composite regularly access the database. To reduce load on the database and to provide faster access to database data, Company X integrates a Coherence adapter into the composite. The Coherence adapter initially puts the data into the cache after it reads from the database the first time.

A Coherence cache is a collection of data objects that serves as an intermediary between the database and client applications. Database data can be loaded into a cache and made available to different applications. The Coherence adapter enables you to perform useful Coherence operations such as adding an item to Coherence cache, obtaining an item, removing an item, and querying items.

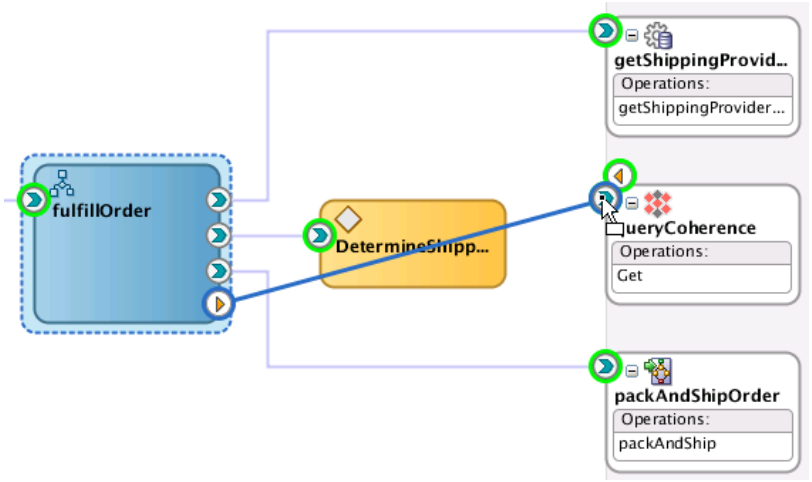
Company X drags a **Coherence** icon from the **Technology** section of the Components window into the **External References** swimlane of the composite. The Adapter Configuration Wizard guides you through configuring the following Coherence adapter details.

- The JNDI name of the Coherence connection.
- The operation to perform. In this case, a get operation is selected to get data from Coherence cache.
- The cache type (XML).
- The cache name.
- The key type (string) (Populated later with the **jca.coherence.Key** property in an invoke activity.).

- The database lookup response schema file (Stores the response from the database lookup directly in Coherence cache.).

When adapter configuration is complete, the **fulfillOrder** BPEL process is connected to the Coherence adapter, as shown in Figure 7-18.

Figure 7-18 fulfillOrder BPEL Process Connected to Coherence Adapter

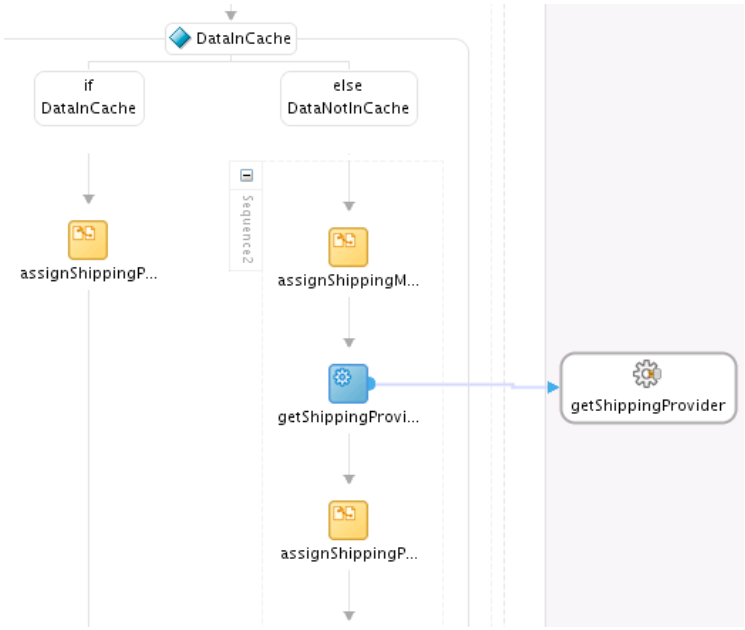


To complete configuration, Company X adds and configures the necessary activities in the **fulfillOrder** process:

- An invoke activity is configured as follows:
  - Invokes the **QueryCoherence** partner link.
  - Defines input and output variables.
  - Assigns the **jca.coherence.Key** property to the **shippingMethodID** value (under the **Properties** tab). This value is sent as the input variable upon invocation of the adapter and eliminates the need to create an assign activity statement in which an input value is assigned to the Coherence query invocation.
- An if activity is configured as follows:
  - The if part assigns the shipping provider name from the Coherence query output variable to the shipping provider name in the process input variable (in the shipping element). This is used later for the **PackAndShipService** designed in Packing and Shipping Orders .
  - An else part includes the **getShippingProvider** database adapter invocation and its two assign activities.

When complete, configuration looks as shown in Figure 7-19.

Figure 7-19 If Statement



Copying the Database Adapter Response into Coherence Cache

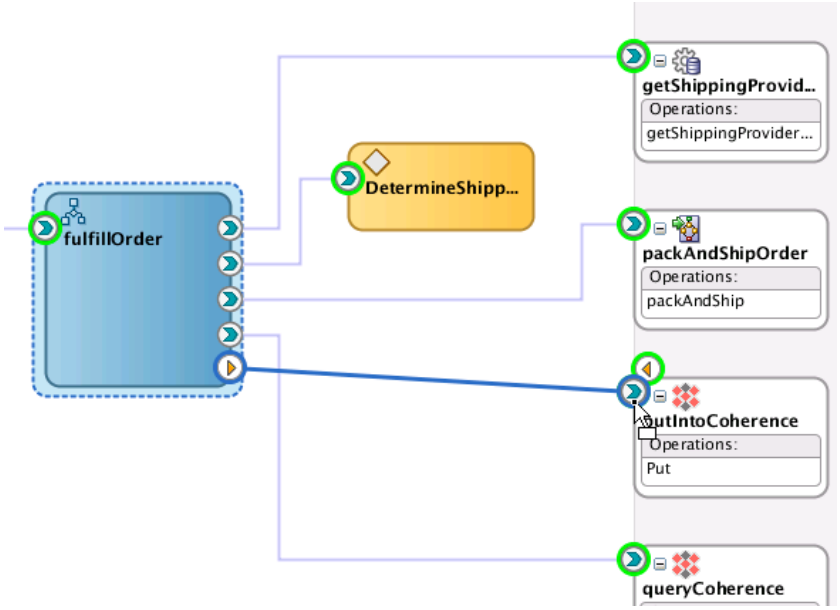
Company X adds another Coherence adapter to the **External References** swimlane of the composite. This adapter copies the database adapter response into Coherence cache so that the shipping provider is available in cache the next time it is looked up.

The Coherence adapter is configured as follows:

- The JNDI name of the Coherence connection.
- The operation to perform. In this case, a put operation is selected to put an item into cache.
- The cache type (XML).
- The cache name.
- The key type (string).
- The key value of **shippingMethodKey**.
- The database lookup response schema file.

To complete configuration, Company X connects the **fulfillment** process to the new Coherence adapter, as shown in Figure 7-20.

Figure 7-20 BPEL Process Connected to Second Coherence Adapter



To complete configuration, Company X adds the necessary activities to the **fulfillOrder** process:

- An invoke activity is configured as follows:
  - Invokes the **putIntoCoherence** partner link.
  - Defines input and output variables.
  - Assigns the **jca.coherence.Key** property to the **shippingMethodID** value (under the **Properties** tab). This value is sent as the input variable upon invocation of the adapter and eliminates the need to create an assign activity statement in which an input value is assigned to the Coherence query invocation.
- An assign activity populates the input variable of the Coherence adapter with the output variable of the database call.

When configuration is complete, the **fulfillment** process looks as shown in Figure 7-21.

Figure 7-21 fulfillment BPEL Process Contents

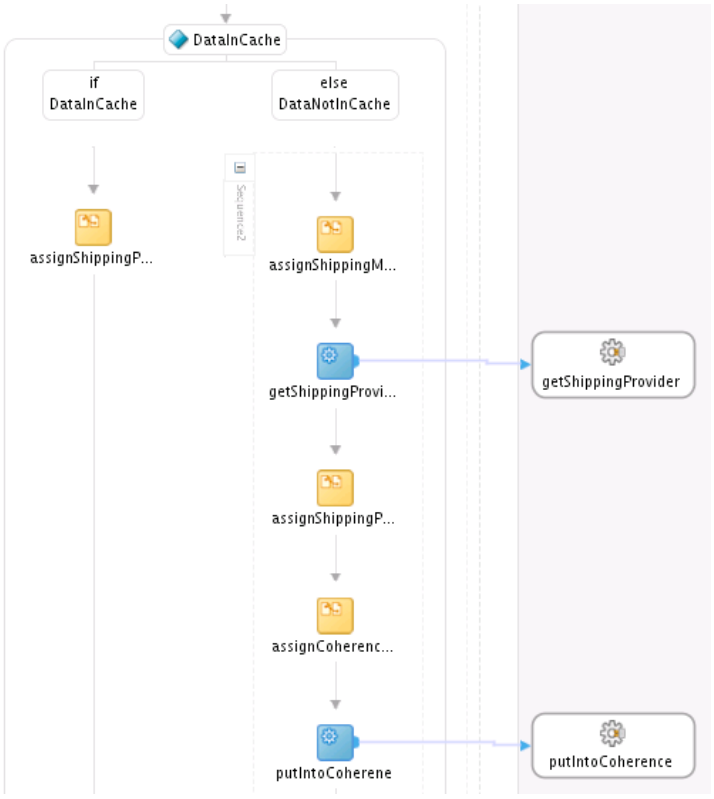
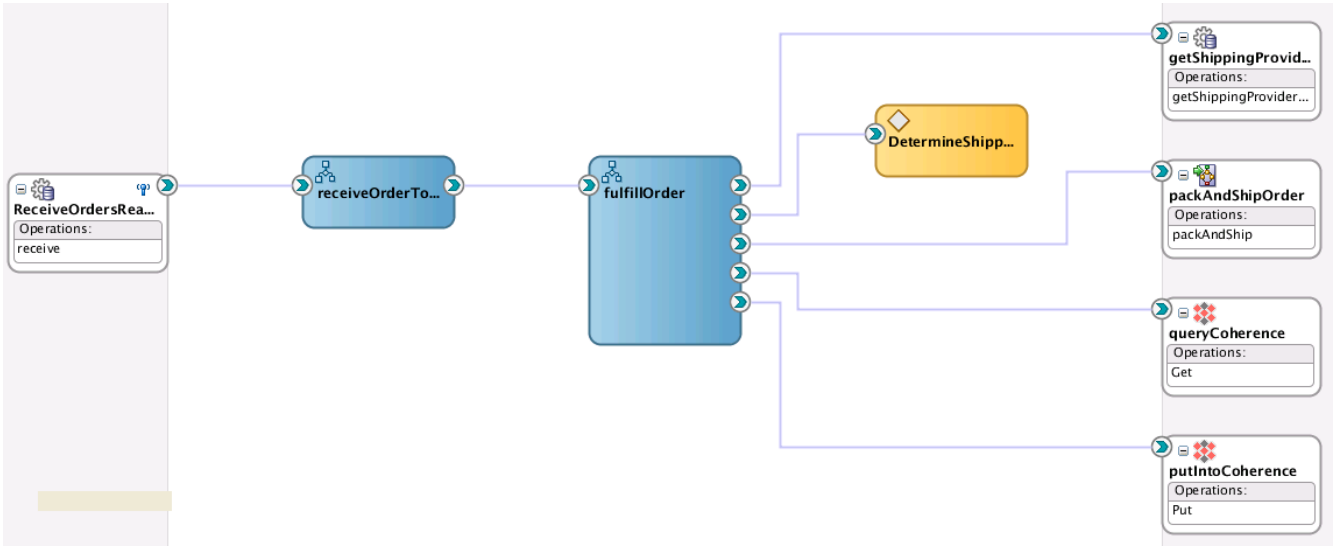


Figure 7-22 shows how this completed business solution appears in the SOA Composite Editor.

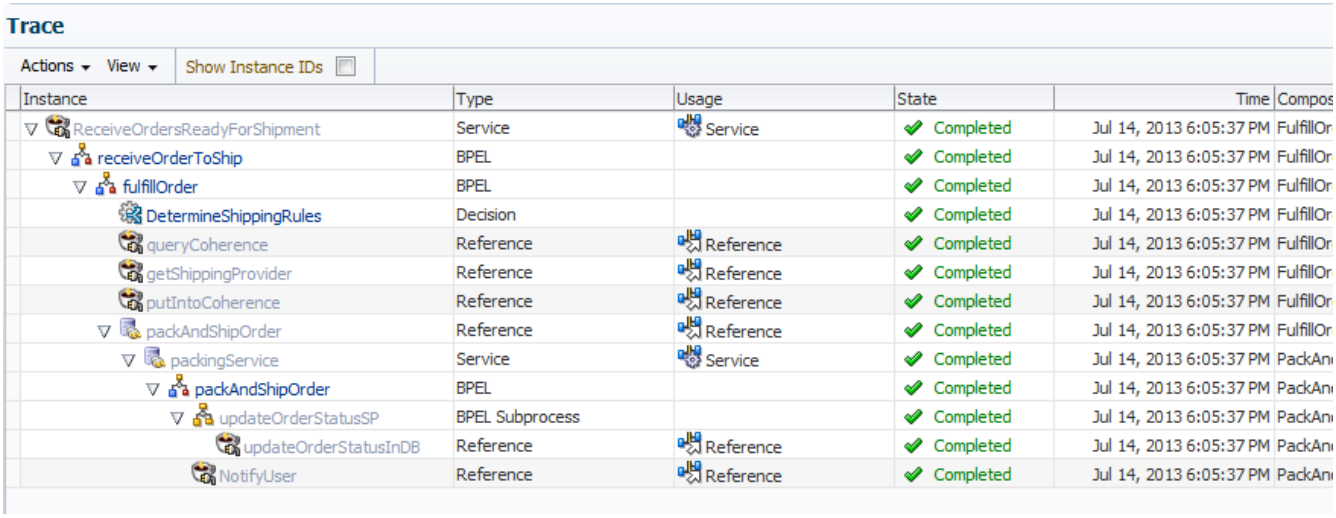
Figure 7-22 Completed SOA Composite Application



Deploying the Composite and Testing the Coherence Adapters

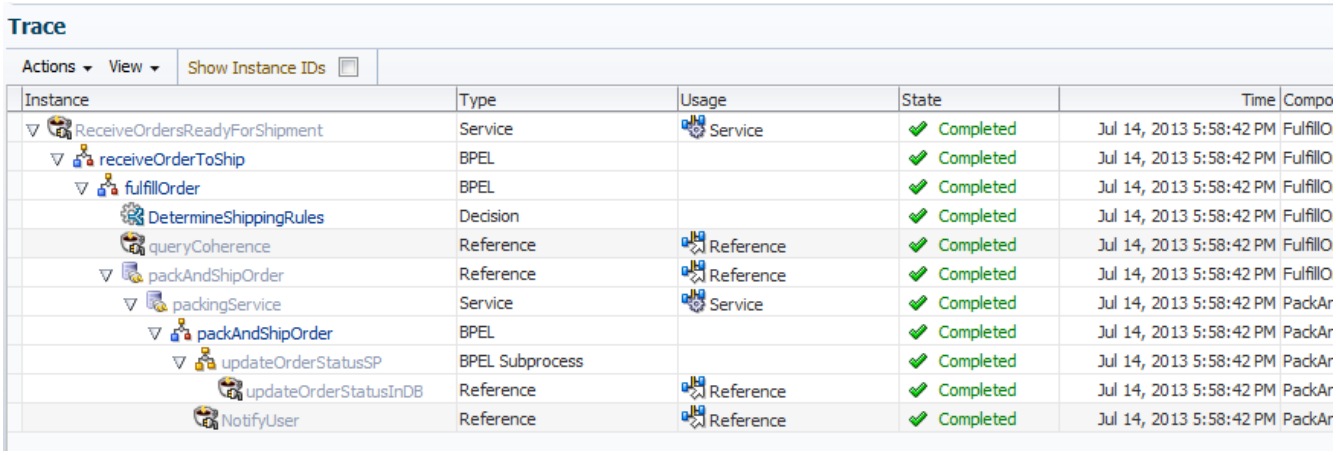
Initially both Coherence adapters are executed. The **queryCoherence** reference returns an empty variable because nothing has been put into Coherence cache yet. The **putCoherence** reference then puts the content into Coherence cache. Figure 7-23 provides details.

Figure 7-23 Flow Trace



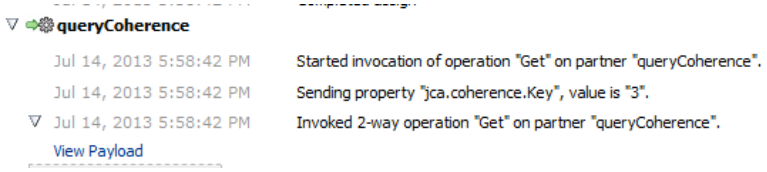
Once the message has been put into Coherence cache, the audit trail changes. Figure 7-24 provides details.

Figure 7-24 Flow Trace with Changes



The **queryCoherence** reference returns the shipping method variable and the database does not need to be queried again. Figure 7-25 provides details.

Figure 7-25 Shipping Method Variable Returned



Related Documentation

Table 7-2 provides references to documentation that more specifically describes the components and features described in this chapter.

Table 7-2 Related Documentation

For Information About...	See...
Adding a project template	Oracle SOA Suite Templates and Reusable Subprocesses in *Developing SOA Applications with Oracle SOA Suite*
Designing a business rule	Overview of Oracle Business Rules in *Designing Business Rules with Oracle Business Process Management* Getting Started with Oracle Business Rules in *Developing SOA Applications with Oracle SOA Suite* Working with Decision Tables in *Designing Business Rules with Oracle Business Process Management*
Creating a composite sensor	Defining Composite Sensors in *Developing SOA Applications with Oracle SOA Suite*
Adding a REST reference	Integrating REST Operations in SOA Composite Applications in *Developing SOA Applications with Oracle SOA Suite*
Configuring Coherence adapters	Oracle JCA Adapter for Coherence in *Understanding Technology Adapters*

8

Scheduling Composite Execution

This chapter describes how Oracle SOA Suite addresses the business challenge of maintaining sufficient inventory levels. Overviews of how key SOA composite application components are created and address this challenge are provided, including Oracle Enterprise Scheduler.
This chapter includes the following sections:

Business Challenge

Company X faces the business challenge of maintaining sufficient inventory. As orders are processed, the stock of items in the warehouse is reduced and inventory must be restocked. For each category of products ordered, Company X has a different supplier. At the end of each day, a separate report must be run for each supplier to identify the number of items for each product to be ordered to return the inventory to its original level.

Business Solution

To address these business challenges, Company X designs a business solution that uses the components described in Table 8-1.

Table 8-1 Components That Provide the Business Solution

Component	How This Component Addresses The Business Challenge	Component Description
SOA composite application	A query inventory composite identifies the total number of items of each product ordered for the day for a given category. This query identifies how much inventory has been reduced during the day.	See Table 3-1 for a description of SOA composite applications.
Oracle Enterprise Scheduler job	Oracle Enterprise Scheduler is used to define a web service job for the query inventory composite and then submit the job with a schedule to run at a specified time.	Oracle Enterprise Scheduler enables you to manage job requests, define metadata, and schedule jobs in Oracle Enterprise Manager Fusion Middleware Control.
Oracle Enterprise Scheduler adapter activation and deactivation	Oracle Enterprise Scheduler activates and deactivates the database adapter in the **fulfillment** service created in Fulfilling Orders using recurring schedules for the activation and deactivation.	Oracle Enterprise Scheduler enables you to schedule adapters in composites to activate and deactivate at specified times.

Subsequent sections of this chapter provide more specific details about how the components in Table 8-1 are used to address the credit validation business challenge.

- Creating a Web Service Job Definition
- Submitting a Job Request on a Schedule
- Applying Schedules to Adapters

Creating a Web Service Job Definition

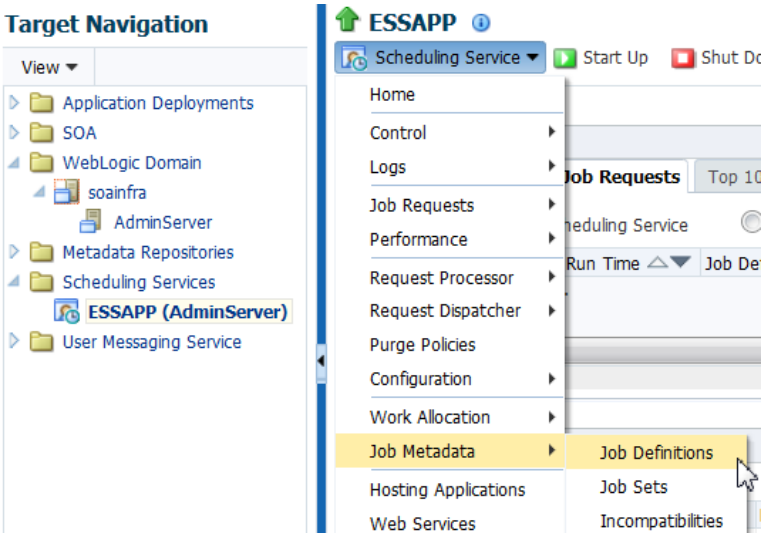
To address the requirement for identifying and maintaining sufficient inventory levels, Company X uses Oracle Enterprise Scheduler.

Oracle Enterprise Scheduler enables you to defer larger transactions to run as jobs at a later time or automate the running of application maintenance work based on a defined schedule. Oracle Enterprise Scheduler enables you to run different job types such as Java, PL/SQL, binary scripts, web services, and EJBs distributed across the nodes in an Oracle WebLogic Server cluster. Oracle Enterprise Scheduler runs jobs securely, providing for high availability, scalability, and load balancing. Oracle Enterprise Scheduler provides monitoring and management through Oracle Enterprise Manager Fusion Middleware Control.

Company X creates a job definition for the **queryInventory** composite that queries their inventory. The composite includes a synchronous BPEL process and a web service as the service binding component. You associate request-specific metadata as job definitions.

Company X creates the job definition by selecting **ESSAPP** in the Oracle Enterprise Manager Fusion Middleware Control navigator, then selecting **Job Metadata** > **Job Definitions** from the **Scheduling Service** menu. Figure 8-1 provides details.

Figure 8-1 Job Definition Creation

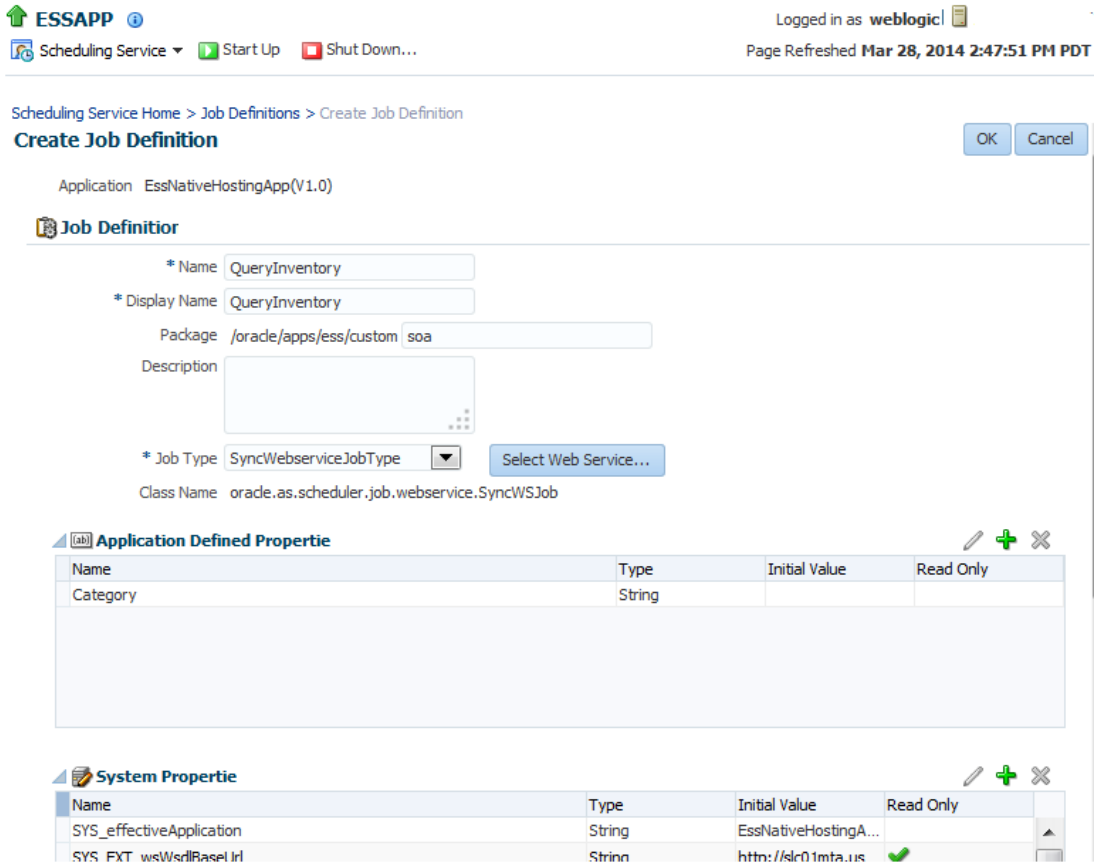


Company X configures a job definition with the following details:

- Job name and display name of **QueryInventory**
- Package value of **soa**
- Job type of **syncWebServiceJobType**

When complete, configuration looks as shown in Figure 8-2.

Figure 8-2 Job Definition Configuration

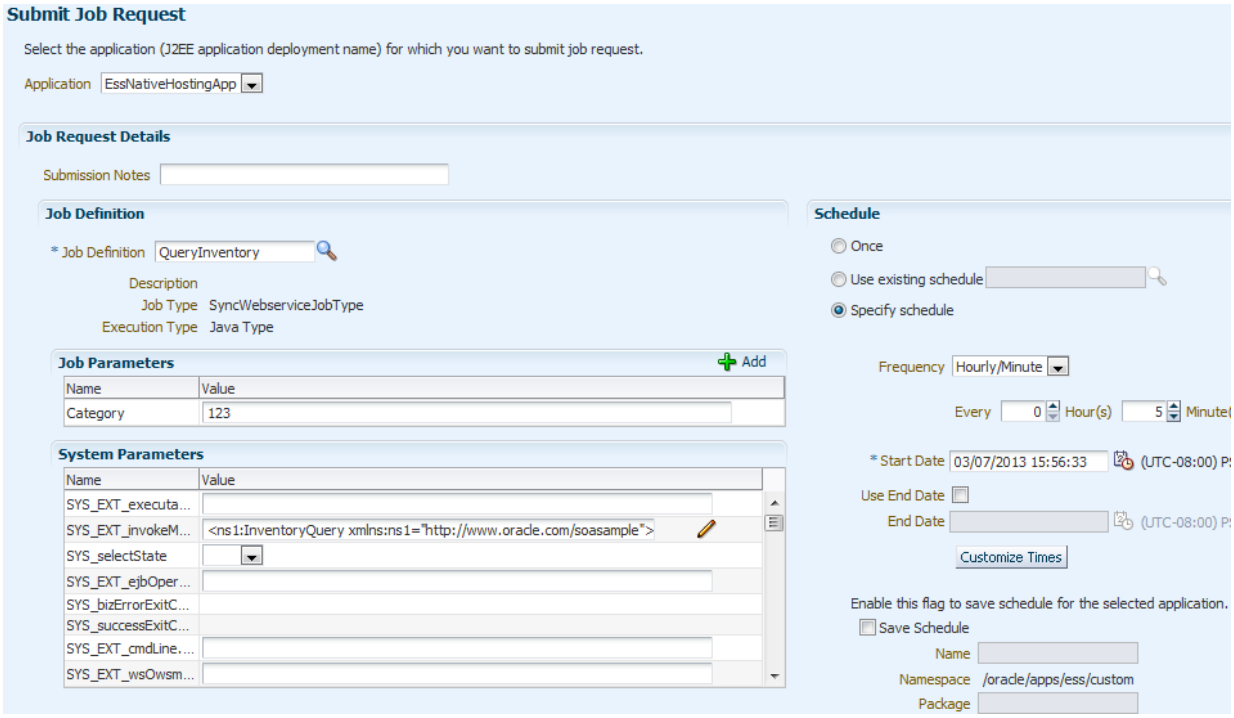


Submitting a Job Request on a Schedule

Company X creates a schedule for running the job definition created in Creating a Web Service Job Definition. A schedule determines when the job runs.

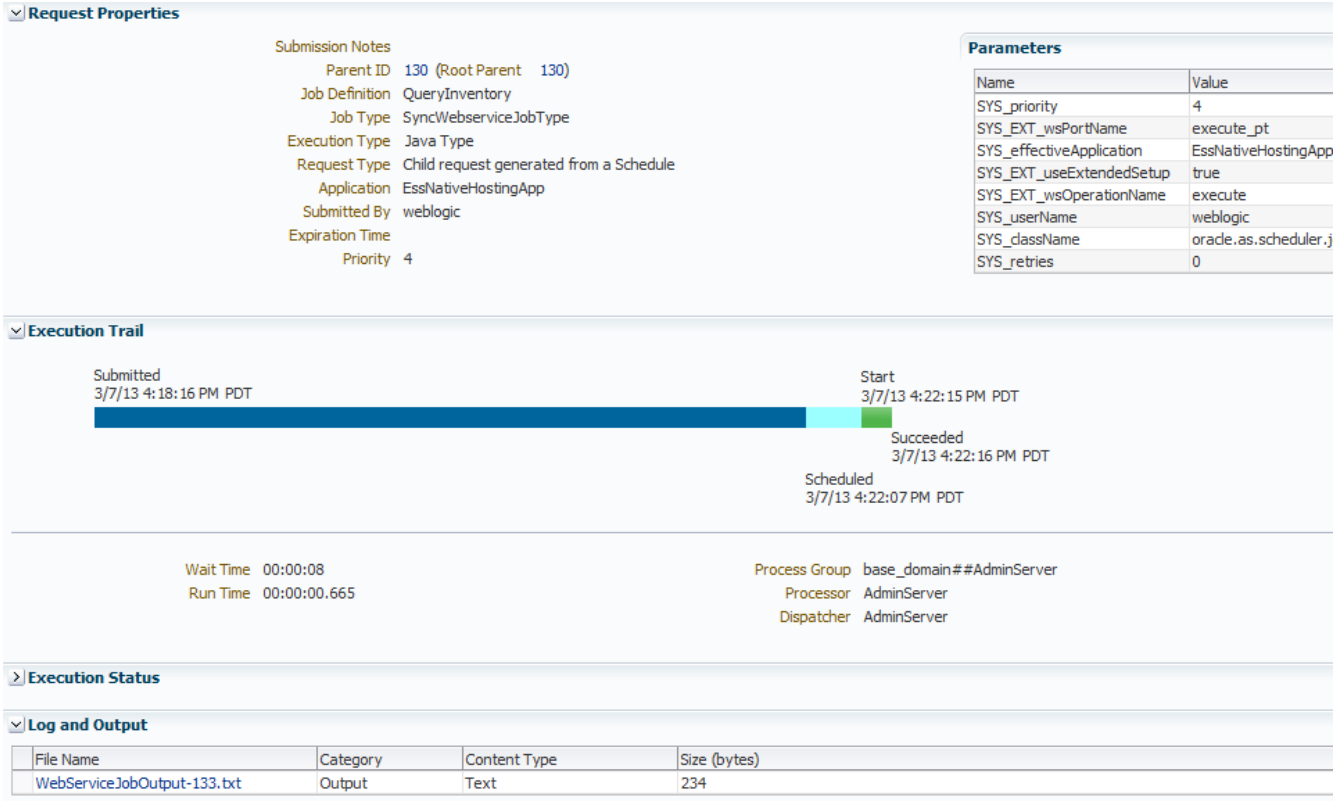
From the **Scheduling Services** menu in the Oracle Enterprise Manager Fusion Middleware Control navigator, Company X selects **Job Requests** > **Submit Job Request**. Company X selects the **QueryInventory** job definition and defines a schedule for running the job. Figure 8-3 provides details.

Figure 8-3 Job Request Submittal



After the job runs, you can review the output. Company X selects **Job Requests** > **Search Job Request** from the **Scheduling Services** menu and navigates to the Request Details page. In the **Log and Output** section at the bottom of the page, details about the web service response are available in an XML file. Figure 8-4 provides details.

Figure 8-4 Output Results of Web Service Response



Applying Schedules to Adapters

Oracle Enterprise Scheduler also enables you to schedule adapters in composites to be activated and deactivated at specified times. You can schedule to activate an adapter during periods when load on the system is minimal. The **fulfillment** composite designed in Fulfilling Orders includes a database adapter as a service input.

Company X uses Oracle Enterprise Scheduler to activate and deactivate the database adapter using recurring schedules. Company X selects **Job Requests** > **Define Schedules** from the **Scheduling Services** list. Company X configures activation and deactivation job definitions for the database adapter with the details shown in Table 8-2. The database adapter is configured to active every ten minutes, and then deactivate every ten minutes.

Table 8-2 Configuration of Activation and Deactivation of Adapters

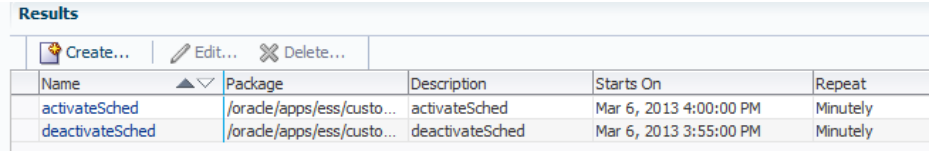
Element	Activation of Adapter	Deactivation of Adapter
Name	activateSched	deactivateSched
Display Name	activateSched	deactivateSched
Package	soa	soa
Frequency	Hourly/Minute	Hourly/Minute

Table 8-2 (Cont.) Configuration of Activation and Deactivation of Adapters

Element	Activation of Adapter	Deactivation of Adapter
Every	10 minutes	10 minutes Deactivation occurs at 5 minutes, 15 minutes, 25 minutes, and so on. Activation occurs at 10 minutes, 20 minutes, 30 minutes, and so on.
Start Date	Ten minutes from now	Five minutes from now
Use End Date	Unchecked	Unchecked

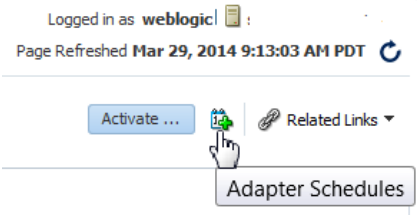
When complete, adapter activation and deactivation configuration looks as shown in Figure 8-5.

Figure 8-5 Activate and Deactivate Adapter Jobs



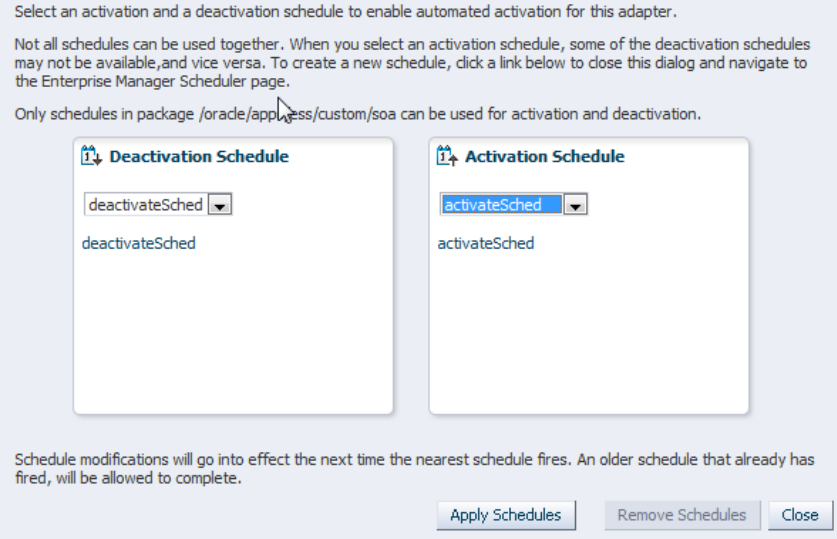
Company X goes to the home page of the database adapter in Oracle Enterprise Manager Fusion Middleware Control and selects to activate and deactivate the database adapter. Figure 8-6 provides details.

Figure 8-6 Database Adapter Activation and Deactivation on Database Adapter Home Page



Company X selects to activate and deactivate the database adapter at the specified times. Figure 8-7 provides details.

Figure 8-7 Adapter Activation Schedule



Related Documentation

Table 8-3 provides references to documentation that more specifically describes the components and features described in this chapter.

Table 8-3 Related Documentation

For Information About...	See...
Creating a job definition and schedule	Managing Oracle Enterprise Scheduler Requests in *Administering Oracle Enterprise Scheduler*
Activating and deactivating adapters	Scheduling JCA Adapter Endpoint Activation and Deactivation using Oracle Enterprise Scheduler in *Administering Oracle SOA Suite and Oracle Business Process Management Suite*

9

Managing File Transfers

This chapter describes Oracle Managed File Transfer (MFT) usage and provides an overview of how Company X integrates its SOA composite applications with Oracle MFT.
This chapter includes the following sections:

- Business Challenge
- Business Solution
- Related Documentation

Business Challenge

Company X has a requirement for its composites to interact with different endpoint types, amongst them file- and FTP-based endpoint types. For example, Company X must be able to use an FTP server to write files to a file system.

Business Solution

To address these business challenges, Company X designs a business solution that uses the components described in Table 9-1.

Table 9-1 Components That Provide the Business Solution

Component	How This Component Addresses The Business Challenge	Component Description
Oracle MFT transfer	• An Oracle MFT transfer receives files and writes them to the file system using the MFT embedded FTP server. • Oracle MFT invokes an MFT service in a SOA composite application and dynamically decides based on file size whether to pass the content inline or by reference.	Oracle MFT is a high performance, standards-based, end-to-end managed file gateway. Oracle MFT provides design, deployment, and monitoring of file transfers using a lightweight web-based, design-time console that includes file encryption, scheduling, and embedded FTP and sFTP servers.

Subsequent sections of this chapter provide more specific details about how the components in Table 9-1 are used to address the file transfer business challenge.

- Creating Transfers, Sources, and Targets
- Creating a SOA Composite Application with an MFT Service
- Sending the Order File to a SOA Composite
- Processing Payload Types
- Invoking the ProcessOrder Composite with an Inline Payload

Creating Transfers, Sources, and Targets

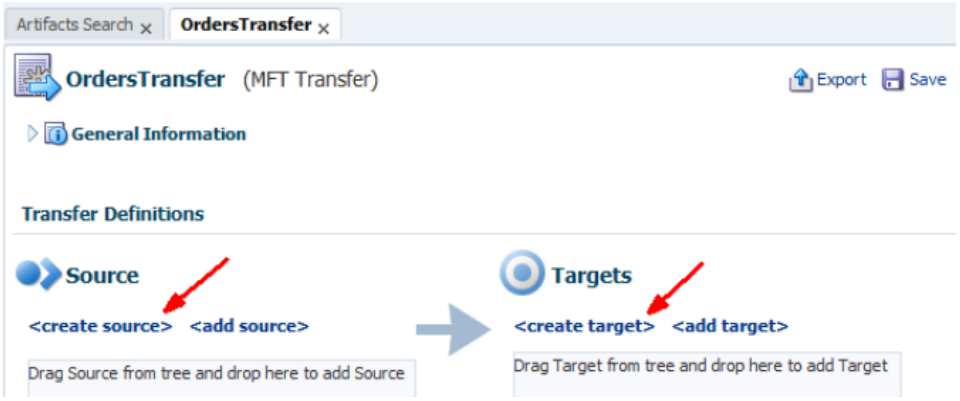
The file delivery structure of Oracle MFT consists of the following types of artifacts:

- Sources define the origin of files.
- Targets define the destination of files.
- Transfers associate a source with one or more targets.

You create these artifacts on the designer page of the Oracle Managed File Transfer Console.

Company X creates a transfer named **OrdersTransfer**, as shown in Figure 9-1. The page includes options for creating sources and targets.

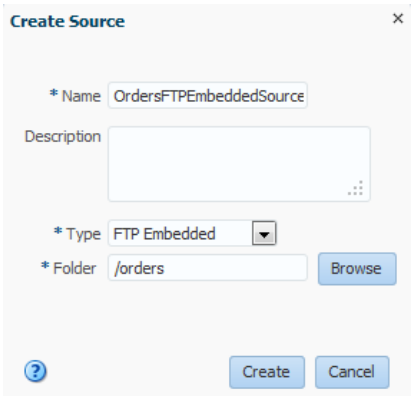
Figure 9-1 Transfer Creation



Company X creates the following source and target artifacts:

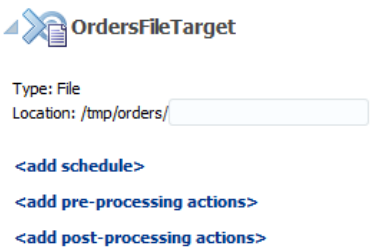
- A source named **CreateSource** in which files are placed in the **/orders** directory. The FTP embedded server is selected for transferring the file. Figure 9-2 provides details. The source location is further defined to process files that adhere to the pattern of **Order*.xml**.

Figure 9-2 Source Location



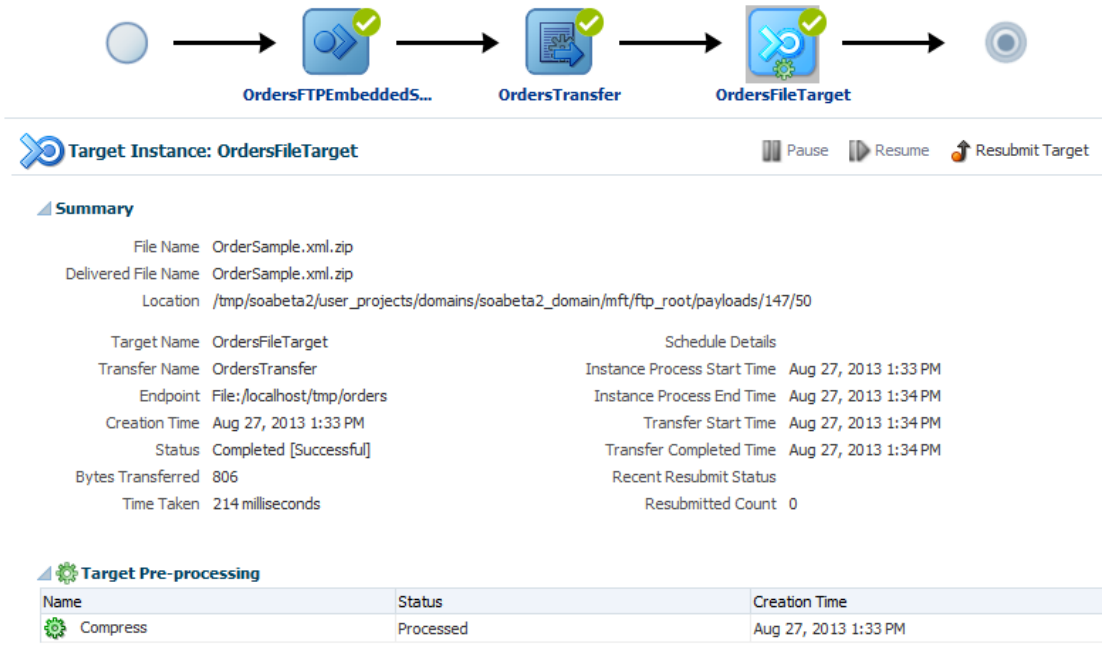
- A target named **OrdersFileTarget** in which transferred files are placed in the **/tmp/orders** directory. File targets can be reused by overriding the location with a subdirectory such as `/tmp/orders/output`. The target is further defined to select a compression level. Figure 9-3 provides details.

Figure 9-3 Target Location



After deployment, files that adhere to the pattern of **Orders*.xml** in **/orders** are written to **/tmp/orders** in compressed format. The Oracle Managed File Transfer Console in Figure 9-4 shows the transfer results.

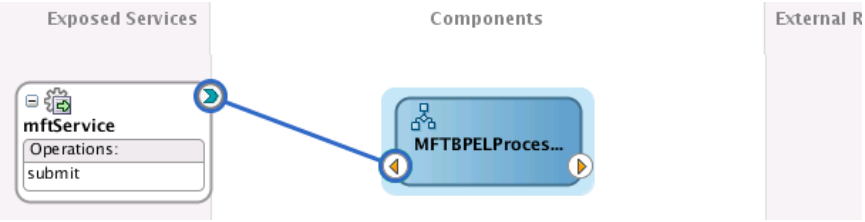
Figure 9-4 Transfer Results



Creating a SOA Composite Application with an MFT Service

Company X creates a SOA composite named **MFTProcessor** and an empty BPEL process by selecting **Define Service Later** in the Create BPEL Process dialog. In the SOA Composite Editor, Company X creates an Oracle MFT service by dragging an **MFT** icon from the **Technology** section of the Components window into the **Exposed Services** swimlane of the composite. The Oracle MFT service is designed to dynamically decide based on file size whether to pass file content inline or by reference. After configuration is complete, the service is wired to the composite, as shown in Figure 9-5.

Figure 9-5 Oracle MFT Service Connected to a BPEL Process

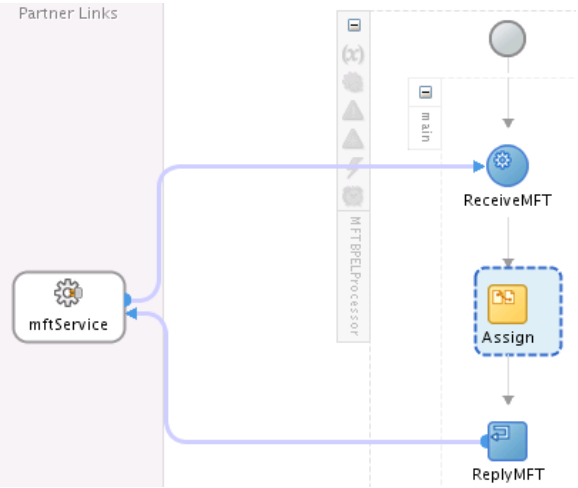


To complete configuration, Company X adds and configures the necessary activities in the **MFTBPELProcess** BPEL process:

- A receive activity is invoked by the Oracle MFT service and is configured with an input variable.
- An assign activity sends the payload type as a response.
- A reply activity returns an output variable to the Oracle MFT service.

When complete, BPEL process design looks as shown in Figure 9-6.

Figure 9-6 BPEL Process with MFT Service

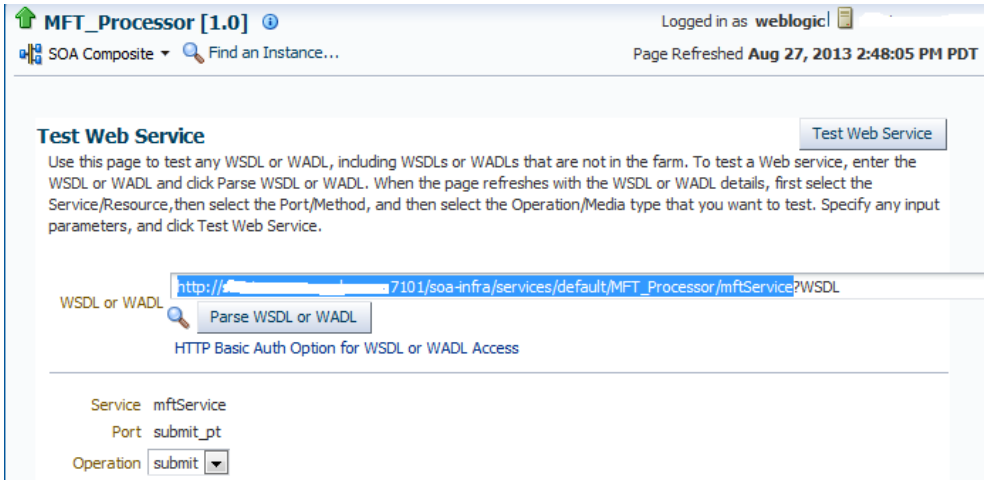


The composite is deployed and verified in Oracle Enterprise Manager Fusion Middleware Control.

Sending the Order File to a SOA Composite

Company X creates an additional Oracle MFT target to invoke the deployed **MFTProcessor** composite. Company X copies the endpoint URL of the **MFTProcessor** composite from the Test Web Service page in Oracle Enterprise Manager Fusion Middleware Control. Figure 9-7 provides details.

Figure 9-7 Copy of Endpoint URL of the Deployed Composite

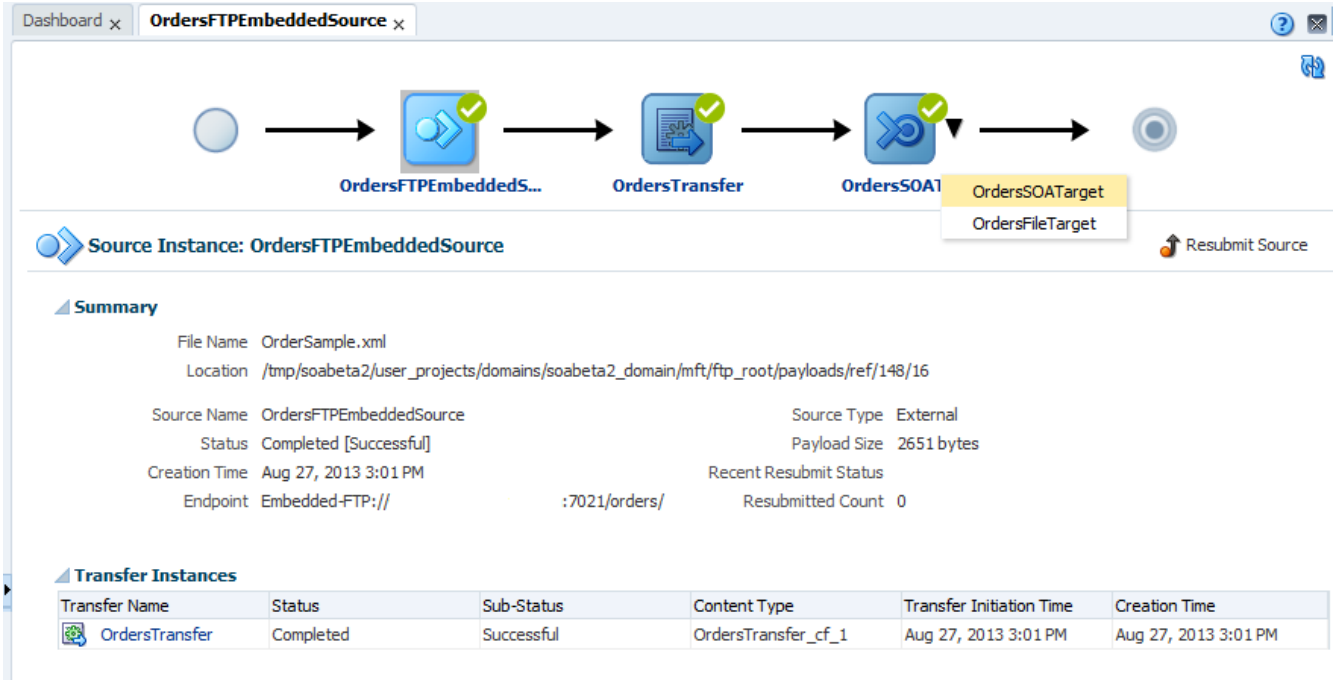


In the Oracle Managed File Transfer Console, Company X creates an **OrdersSOATarget** target with the URL of the composite copied from the Test Web Service page and selects inline attachment as the file delivery method. Figure 9-8 provides details.

Figure 9-8 MFT Target

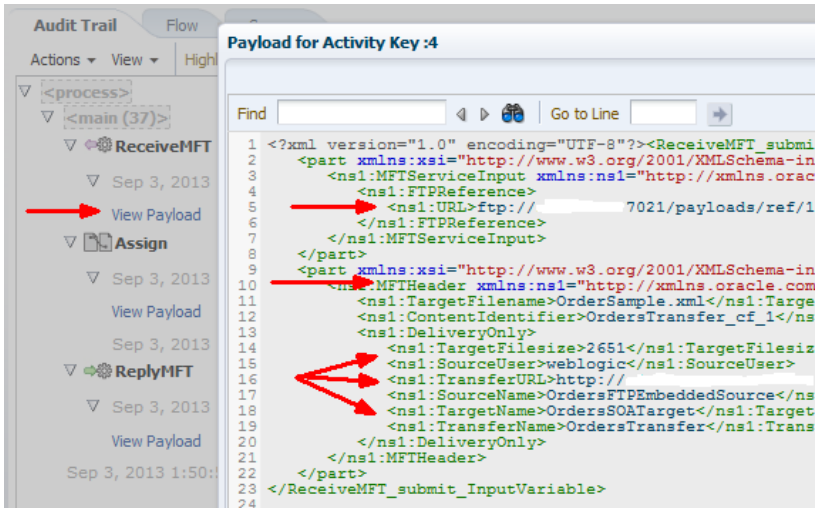
After redeployment, files that adhere to the pattern of **Orders*.xml** in the **/orders** directory are transferred to the **/tmp/orders** directory. The Oracle Managed File Transfer Console shows the transfer results in Figure 9-9. The **OrdersSOATarget** target is displayed along with the previously created source **OrdersFileTarget**.

Figure 9-9 Output Results



The **View Payload** link in the audit trail of the business flow instance of the composite in Oracle Enterprise Manager Fusion Middleware Control shows that the file was sent as an FTP reference pointing back to the Oracle MFT embedded FTP server. Figure 9-10 provides details.

Figure 9-10 Audit Trail

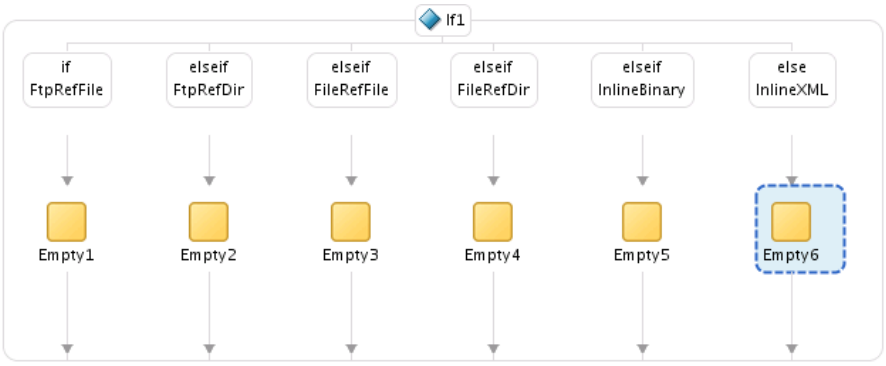


Processing Payload Types

Company X adds an if activity to the BPEL process of the **MFTProcessor** composite. Each if activity branch defines an XPath expression for processing a different type of payload that Oracle MFT sends to a SOA composite and implements inline processing to call the

ProcessOrder service created in Creating an Order Processing System. Figure 9-11 provides details.

Figure 9-11 If Activity Branches



The branches are defined as follows:

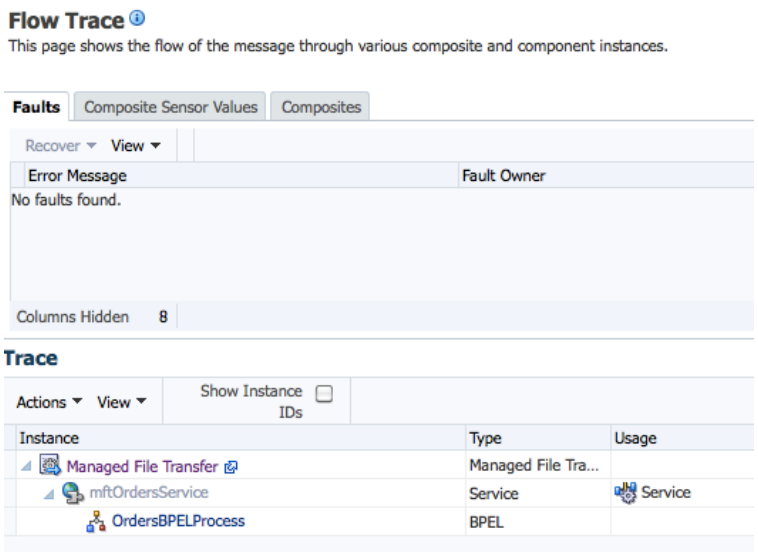
- **FtpRefFile**: Passes a file by reference. The file is transferred using FTP. This branch uses an XPath expression of `xp20:compare($PayloadType,'FtpRefFile') = 0`.
- **FtpRefDir**: Passes a directory of files by reference. The directory of files is transferred using FTP. This branch uses an XPath expression of `xp20:compare($PayloadType,'FtpRefDir') = 0`.
- **FileRefFile**: Passes a file by reference. The file is passed requiring special handling using a file adapter. This branch uses an XPath expression of `xp20:compare($PayloadType,'FileRefFile') = 0`.
- **FileRefDir**: Passes a directory of files by reference. A directory of files is passed requiring special handling using a file adapter. This branch uses an XPath expression of `xp20:compare($PayloadType,'FileRefDir') = 0`.
- **InlineBinary**: Passes a binary file inline. This branch uses an XPath expression of `xp20:compare($PayloadType,'InlineBinary') = 0`.
- **InlineXML**: Passes an XML file inline. Because this is the else branch, no expression is required.

After redeployment, the audit trail of the business flow instance shows two order files (**OrderSample.xml** and **OrderSampleLarge.xml**). Each file takes a different path in the SOA BPEL process based on the XPath expression in the if branch. Figure 9-12 provides details.

Figure 9-12 Audit Trail

In the **Trace** table of the Flow Trace page, there is a link next to the **Managed File Transfer** name that takes you to the Oracle Managed File Transfer Console. Figure 9-13 provides details.

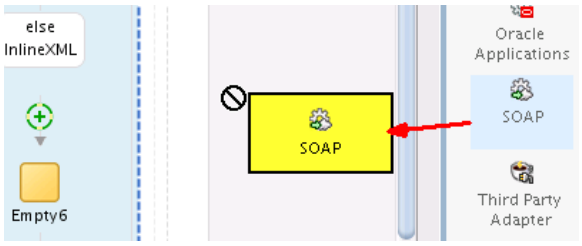
Figure 9-13 Link to Oracle Managed File Transfer Console



Invoking the ProcessOrder Composite with an Inline Payload

Company X drags a **SOAP** icon from the **Technology** section of the Components window into the **External References** swimlane of the **MFTProcessor** composite. The SOAP service reference is configured to invoke the **ProcessOrder** endpoint. Figure 9-14 provides details.

Figure 9-14 SOAP Reference

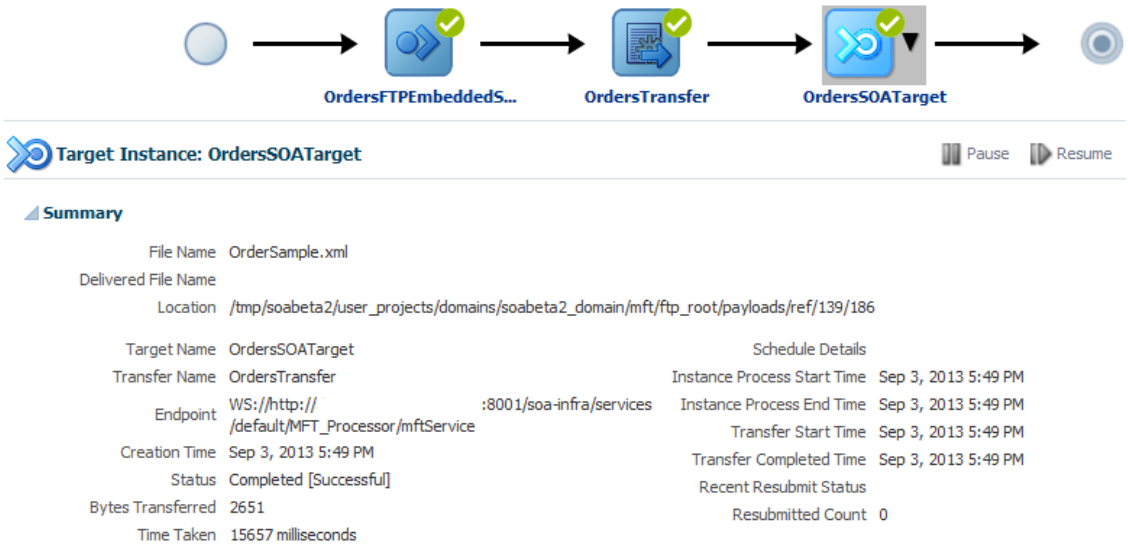


After SOAP configuration is complete, Company X adds and configures the necessary activities in the BPEL process:

- An invoke activity is added to the else part of the if activity to invoke the web service. Input and output variables are created.
- An assign activity maps the MFT service input payload to the **ProcessOrder** input variable.

After redeployment, the **OrdersSample.xml** file is placed in **/orders** and transferred to **/tmp/orders**. The Oracle Managed File Transfer Console shows the transfer results in Figure 9-15.

Figure 9-15 Output File



The audit trail shows the **ProcessOrder** composite was invoked and the order was processed. Figure 9-16 provides details.

Figure 9-16 Audit Trail

Related Documentation

Table 9-2 provides references to documentation that more specifically describes the features described in this chapter.

Table 9-2 Related Documentation

For Information About...	See...
Key Oracle MFT concepts	Understanding Oracle Managed File Transfer in *Using Oracle Managed File Transfer*
Oracle MFT transfers, targets, and sources	Designing Artifacts: Transfers, Sources, and Targets in *Using Oracle Managed File Transfer*

10

Gaining Business Insights with Oracle Business Activity Monitoring

This chapter describes how Oracle SOA Suite addresses the business challenge of gaining business insights into customer order requests. Overviews of how key SOA composite application components address this challenge are provided, including process analytics and Oracle Business Activity Monitoring (BAM).
This chapter includes the following sections:

Business Challenge

Company X has a requirement to gain business insights into the customer ordering system to identify the following trends:

- The number of orders approved, received, or rejected.
- The reasons for rejections, such as insufficient income, credit, or age.
- The number of automatic or manual credit card approvals per state.

Business Solution

To address this business challenge, Company X designs a business solution that uses the components described in Table 10-1.

Table 10-1 Components That Provide the Business Solution

Component	How This Component Addresses The Business Challenge	Component Description
Process analytics	Process analytic measurements and business indicators are defined on activities in a BPEL process.	Process analytics provide a uniform measurement mechanism for collecting disparate data in BPEL processes.
Oracle BAM	When a SOA composite is configured with analytics measurements and business indicators, composite-specific physical and logical data objects are automatically created in Oracle BAM upon the composite's deployment. Oracle BAM Composer is used to create a dashboard of these analytics to gain business insights into customer order requests.	Oracle BAM monitors business processes in real time to help you make informed tactical and strategic business decisions.

Subsequent sections of this chapter provide more specific details about how the components in Table 10-1 are used to address the business insight challenge.

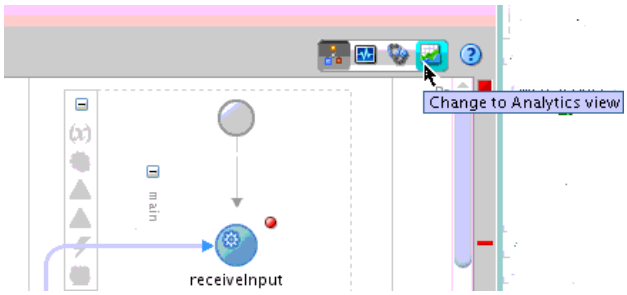
Adding Business Indicators and Measurements to a Composite

Oracle SOA Suite provides a uniform process schema across components such as BPEL processes, Oracle BPM, and human workflow. Oracle BAM models this schema as data objects that are persisted in the Oracle BAM server. These data objects are preseeded and populated by the processes automatically.

Oracle SOA Suite provides a feature called process analytics. This feature enables Company X to specify user metrics such as business indicators and measurements on activities in a BPEL process. When the composite is deployed, corresponding, custom, derived data objects are created and updated in Oracle BAM.

Company X opens a BPEL process in Oracle JDeveloper (for this scenario, named **CCComposite**) in which to specify user metrics and clicks the **Change to Analytics view** icon, as shown in Figure 10-1. This opens the process in analytics view. Analytics view enables you to define business indicators and measurements in the BPEL process.

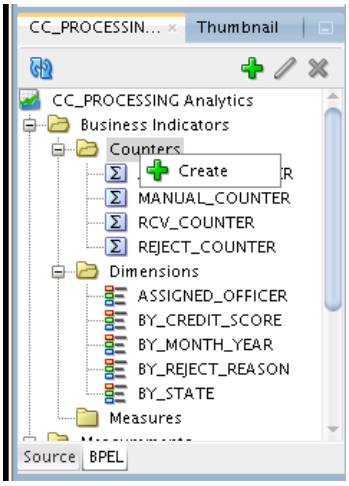
Figure 10-1 Analytics View in a BPEL Process



- Business indicators are defined in the Structure window, and consist of the following:
  - Counters: Track the number of times a process instance completes a marked element such as a BPEL activity.
  - Dimensions: Label group or filter measures.
  - Measures: Store the values of a variable such as a sales amount, an employee salary, and so on.

    Figure 10-2 shows defined business indicators in the Structure window.

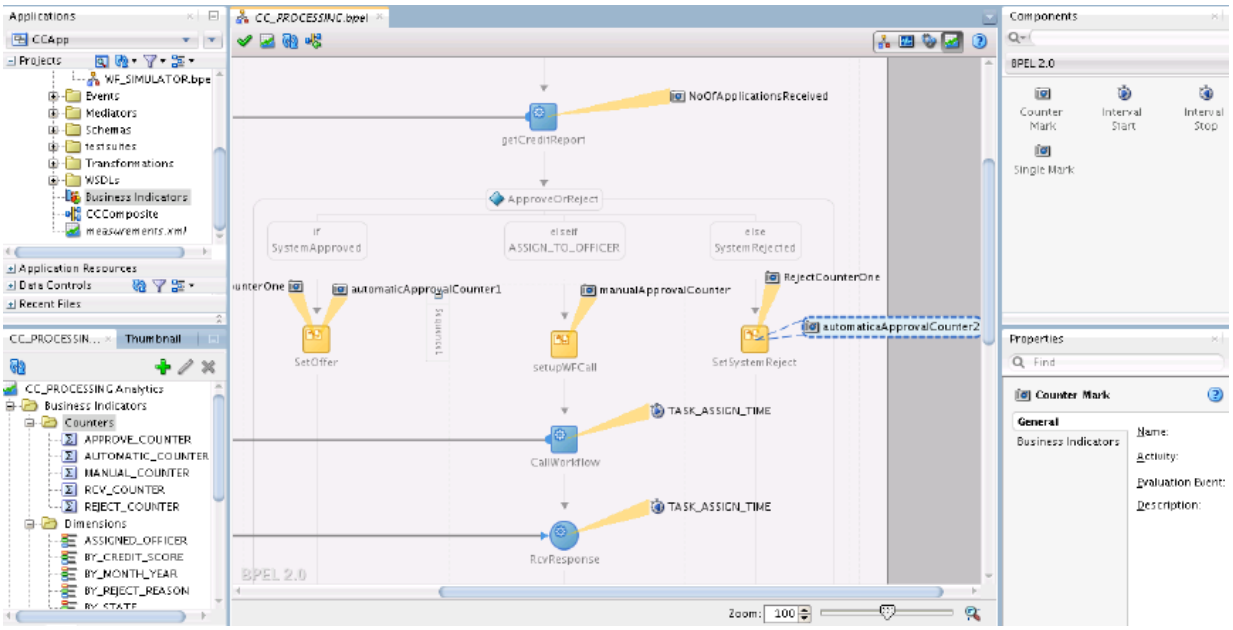
Figure 10-2 Business Indicators Defined in the Structure Window



- Measurements are displayed in the Components window, and are dragged and dropped onto activities for configuration. The following types are available:
  - Counter mark: The BPEL activity on which the counter mark is taken. In this BPEL process, Company X defines counter marks for tracking the number of manual approvals, automatic approvals, and rejected orders.
  - Interval start: The BPEL activity on which the interval starts.
  - Interval stop: The BPEL activity on which the interval ends.
  - Single mark: The BPEL activity on which the single mark is taken.

  Figure 10-3 provides details. Measurements that have been dragged from the Components window onto appropriate BEPL activities and configured are indicated by text annotations. The defined business indicators are displayed in the Structure window.

Figure 10-3 BPEL Process with Defined Business Indicators and Measurement Analytics



When a SOA composite is configured with analytics measurements and business indicators, composite-specific physical and logical data objects are automatically created in Oracle BAM upon the composite's deployment. These composite-specific data objects have the columns for business indicators and are populated with analytics information corresponding to processes in that composite.

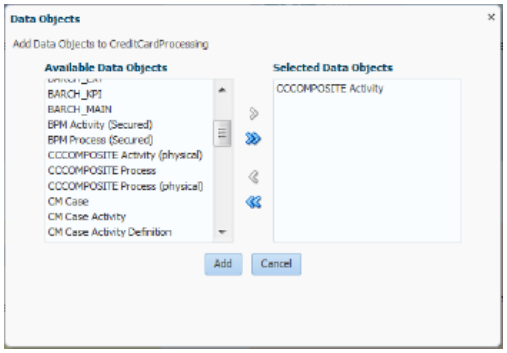
Gaining Business Insights with Oracle BAM Dashboards

Company X deploys the composite, which begins processing order requests.

When Company X specifies business indicators and measurements in a SOA composite application, corresponding, custom, derived data objects are created and updated in Oracle BAM when the composite is deployed. Oracle BAM data objects are populated with appropriate analytics as per the measurements defined in the composite.

Company X logs in to Oracle BAM Composer and creates a new project named **Credit Card Processing** in Oracle BAM Composer. During creation, Company X adds the activity data object corresponding to the deployed **CCComposite** composite into the project, as shown in Figure 10-4. These composite-specific data objects have the columns for business indicators and are populated with analytics information corresponding to processes in that SOA composite.

Figure 10-4 Activity Data Object Selection



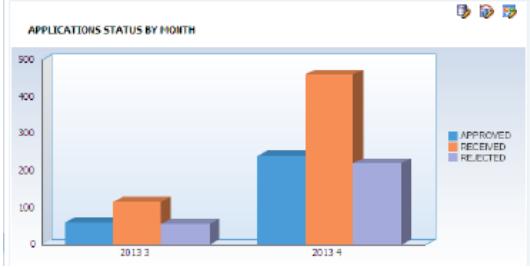
Company X then creates a dashboard with a display name of **Credit Card Processing** in the project. Oracle BAM provides a rich set of ready-to-use dashboards for all major components of Oracle BPM Suite and Oracle SOA Suite. These dashboards include metrics such as task queue depth and bottleneck analysis of business processes.

Company X designs a bar graph with three different counters for displaying the approved, received, and rejected applications by month.

- A display name of **APPLICATIONS STATUS BY MONTH**.
- A business view query modeled with measurements and dimensions that Company X designed in the **CCComposite** composite in Oracle JDeveloper.
  - A data object of **/oracle/processanalytics/CCComposite Activity**.
  - A counter for **APPROVE_COUNTER BI** (Display name is changed to **APPROVED**).
  - A counter for **RCV_COUNTER BI** (Display name is changed to **RECEIVED**).
  - A counter for **REJECT_COUNTER BI** (Display name is changed to **REJECTED**).
  - A time dimension of one month.

Figure 10-5 provides details.

Figure 10-5 Bar Graph for Displaying the Approved, Received, and Rejected Applications by Month

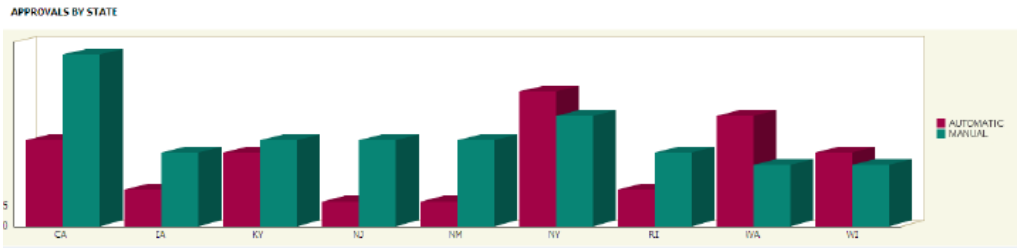


Company X designs a second bar graph for tracking the number of manual and automatic approvals by state.

- A display name of **APPROVALS BY STATE**.
- A business view query modeled with measurements and dimensions that Company X created in the **CCComposite** composite in Oracle JDeveloper.
  - A data object of **oracle/processanalytics/CCComposite Activity.**
  - A counter for **AUTOMATIC_COUNTER BI** (Display name is changed to **AUTOMATIC**).
  - A counter for **MANUAL_COUNTER BI** (Display name is changed to **MANUAL**).
  - A dimension based on state.

Figure 10-6 provides details.

Figure 10-6 Second Bar Graph for Tracking the Number of Manual and Automatic Approvals by State



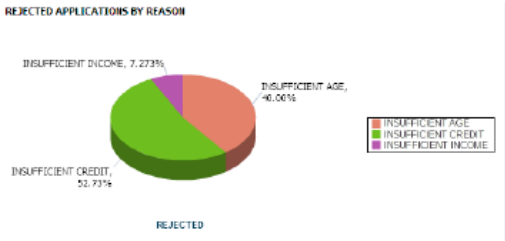
Company X designs a pie graph to the right of the **APPLICATIONS STATUS BY MONTH** graph to track the number of rejected applications due to different reasons.

- A display name of **REJECTED APPLICATIONS BY REASON**.
- A business view query modeled with measurements and dimensions that Company X created in the **CCComposite** composite in Oracle JDeveloper.
  - A data object of **oracle/processanalytics/CCComposite Activity.**
  - A counter for (**REJECT_COUNTER BI**) (Display name is changed to **REJECTED**).
  - A dimension set to **BY REJECT REASON** (due to insufficient age, credit, or income).

– A **Text Type** set to **Text and Percentage** for the pie slice.

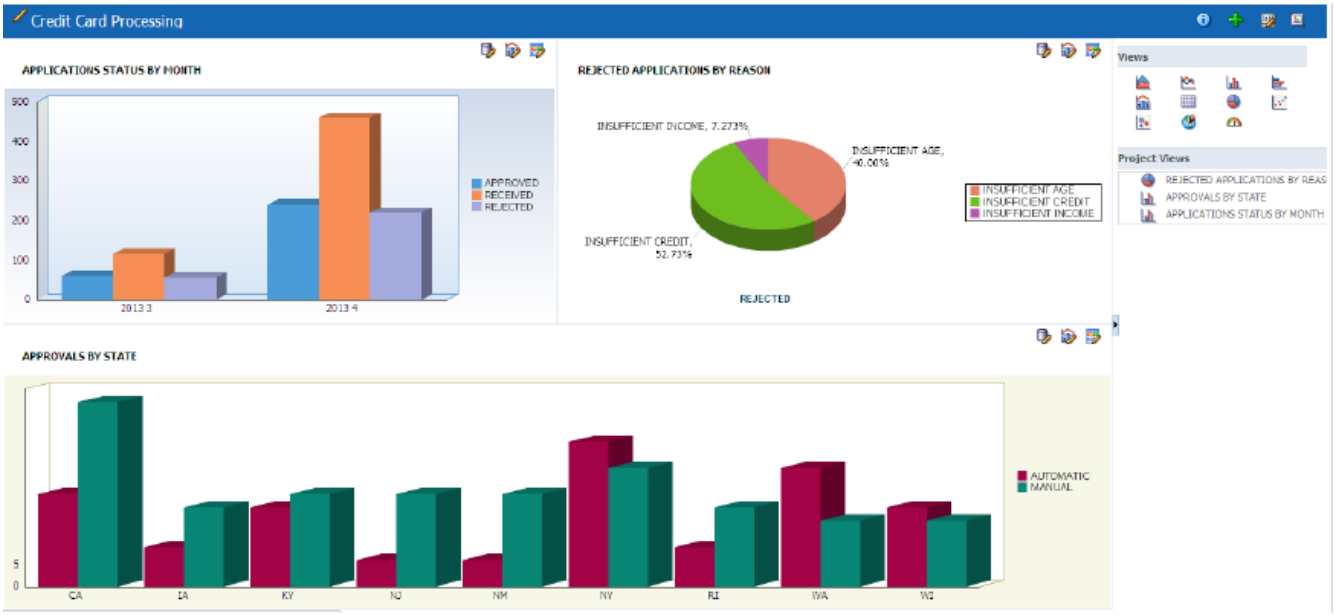
Figure 10-7 provides details.

Figure 10-7 Pie Graph to Track the Number of Rejected Applications Due to Different Reasons



When complete, the entire dashboard looks as shown in Figure 10-8.

Figure 10-8 Oracle BAM Dashboard



Related Documentation

Table 10-2 provides references to documentation that more specifically describes the features described in this chapter.

Table 10-2 Related Documentation

For Information About...	See...
Creating process analytics	Configuring BPEL Process Analytics in *Developing SOA Applications with Oracle SOA Suite*

Table 10-2 (Cont.) Related Documentation

For Information About...	See...
Designing Oracle BAM dashboards	Planning and Creating Projects in *Monitoring Business Activity with Oracle BAM* Creating Dashboards in *Monitoring Business Activity with Oracle BAM* Creating Business Queries in *Monitoring Business Activity with Oracle BAM*