Oracle Applications Integration Cookbook: A Developer’s Guide

An Oracle White Paper
August 2005

Oracle Integration Cookbook: A Developer's Guide

INTRODUCTION

Over the last 20 years, organizations invested heavily in automating specific business tasks and processes. This proved to be a good investment increasing productivity across the board and improving productivity exponentially. However, it did bring up some issues. How do you not just make your business process automated, but optimized? How can I ensure that the data used by all systems is the same and of high quality? How can I instrument my business processes to include relevant contextual information for key decision makers? All at the same time improving the ability for my organization to react and keeping costs low? While there is no one answer for all these questions, one technology helps to address all the points outlined above – Integration. Integration is common requirement across all of these problems.

Oracle E-Business Suite, PeopleSoft and JD Edwards have recognized the importance of integration and have opened up several key integration points into our applications at a business process level, the event level and the API level. IT organizations utilize these efficient and flexible platform-neutral integration points to seamlessly participate in business process orchestration environments, such as Oracle BPEL Process Manager. Business process orchestration takes application integration to the next level by using an application independent rules engine to manage business processes that span multiple applications.

Oracle BPEL Process Manager

Business Process Execution Language for Web Services (BPEL) provides enterprises with an industry standard for business process orchestration and execution. As BPEL is fast emerging as the industry standard for business process orchestration, there is a growing need for integration between Oracle BPEL Process Manager, Oracle E-Business Suite, PeopleSoft and JD Edwards applications. Oracle BPEL

Process Manager enables organizations to model and deploy business processes based on the Business Process Execution Language for Web Services (BPEL) standard. As the cornerstone of process orchestration and execution within a Service-Oriented Architecture platform, the BPEL standard provides an enterprise blueprint for reducing the cost and complexity of integration projects by utilizing standards to improve portability and reuse– while increasing their strategic value.

For more information please visit this web site:

http://www.oracle.com/appserver/bpel_home.html

Integration Types

To fully leverage the power of integration, its imperative that you understand the various types of integration possible. Integrations can be broadly classified into the following categories:

- **Synchronous:** Synchronous Integration is a request response mode of operation between two systems. In short, the calling application expects a response from the provider when a method on the latter is invoked in the same thread. For example, a customer creates a Purchase Order in the E-Business Suite by invoking a Purchase Order Service and the application generates the PO number as a response in real time. This PO number is then used as a reference to complete other related activities.
- **Asynchronous Integrations**: Asynchronous Integration solutions are helpful when enterprise applications take an indeterminate amount of time to respond to a calling application. In this case, the initiating application sends the request and proceeds with its own actions without having to wait for the provider application to process in the same thread. The provider application picks up the request at a later stage and processes the message. For example, in an Order to Cash flow the buyer could put in a request for products and does not have to wait for the supplier to respond right away to perform other steps in the process.
- **Batch Integrations:** Batch integrations allow you to import or export large batches of data that are collected over a period of time into the system. Typical end-of-day transactions or scheduled data synchronization routines utilize this technique. For example, updated customer data collected through various applications in the IT ecosystem is consolidated into a single source as an end of day process.
- **Process Oriented Integrations:** Process Orchestration takes application integration to the next level by using an application independent state engine to manage business processes that span multiple applications. Process Orchestration can involve system-to-system integration, system to human task integration, or human task to human task integration. For example, automating an Order-to-Cash or a new employee process flow. Such a process could involve orchestrating integration across multiple enterprise applications.

How To Use This Document

This document exists to clarify Oracle's recommended integration approach, and to provide application developers with specific standards and guidelines for building Integrations using BPEL Process Manager. This document assumes that you are using the following versions of the product

- **Oracle eBusiness Suite:** 11.5.10 and above
- **PeopleSoft:** Tools Version: 8.46 and above
- **JD Edwards:** Tools version: 8.95 and above
- **Oracle Application Server:** 10.1.2

This technical white paper provides recommended methodology to customers and system integrators. It describes the recommended best practices to show how customers can build process orchestration solutions using the Oracle BPEL Process Manager by leveraging the available Integration technologies that are Synchronous, Asynchronous and Batch-based across all the three application suites (Oracle EBS, PeopleSoft Enterprise and JD Edwards EnterpriseOne). Here is a synopsis of what each chapter will cover:

- Chapter 1 contains important background material on the available Integration options within each Application Suite.
- Chapter 2 provides an overview of how BPEL can be used for implementing Business Process by Orchestrating the Integration across all three Application Suites.
- Chapter 3 provides technical deep dive information on how Oracle E-Business Suite can be integrated with BPEL.
- Chapter 4 provides technical deep dive information on how PeopleSoft Applications can be integrated with BPEL.
- Chapter 5 provides technical deep dive information on JD Edwards Applications can be integrated with BPEL.
- Appendix: Provides Code samples

CHAPTER 1: AVAILABLE TECHNOLOGY INTERFACES

Oracle E-Business Suite

Business Event System

Business Event System (BES) provides the mechanism for customers to build integration with other applications and systems such as Oracle BPEL Process Manager in an easy, non-invasive manner. BES consists of the Event Manager, which lets you register subscriptions to significant events, and event activities, which let you model business events within workflow processes. Subscriptions can include custom logic in both Java and PL/SQL.

XML Gateway

Oracle XML Gateway is a set of services that allows easy integration with the Oracle E-Business Suite to support XML messaging.

Oracle XML Gateway consumes events raised by the Oracle E-Business Suite and subscribes to inbound events for processing. Oracle XML Gateway uses the message propagation feature of Oracle Advanced Queuing to integrate with the Oracle Transport Agent to deliver messages to and receive messages from business partners. Oracle XML Gateway supports both Business-to-Business (B2B) and Application-to-Application (A2A) initiatives. B2B initiatives include communicating business documents and participating in industry exchanges. An example of an A2A initiative is data integration with legacy and disparate systems.

With Oracle XML Gateway services, you are assured consistent XML message implementation when integrating with the Oracle E-Business Suite, thereby lowering integration costs and expediting message implementation while supporting corporate e-business initiatives.

For more information please visit this web site:

http://download-west.oracle.com/docs/cd/B12190_11/current/acrobat/115ecxug.pdf

Oracle e-Commerce / EDI Gateway

Oracle Applications provides users with the ability to conduct business electronically between trading partners based on Electronic Commerce standards and methodology. One form of Electronic Commerce is Electronic Data Interchange (EDI). EDI is an electronic exchange of data between trading partners. Interface data files are exchanged in a standard format to minimize manual effort, speed data processing, and ensure accuracy. The Oracle e-Commerce Gateway (formerly known as Oracle EDI Gateway) performs the following functions:

- Define trading partner groups and trading partner locations.
- Enable transactions for trading partners.
- Provide general code conversion between trading partner codes or standard codes and the codes defined in Oracle Applications.
- Define interface data files so that application data can integrate with your trading partner's application or an EDI translator.

- For inbound transactions, import data into application open interface tables so that application program interfaces (API) can validate and update Oracle application tables.
- For outbound transactions, extract, format, and write application data to interface data files.

Oracle e-Commerce Gateway augments the existing standard paper document capabilities of Oracle Applications, or adds functionality where no corresponding paper documents exist.

For more information please visit this web site:

http://download-west.oracle.com/docs/cd/B12190_11/current/acrobat/115eccug.pdf

Concurrent Program / Interface Tables

In Oracle Applications, concurrent processing simultaneously executes programs running in the background with online operations to fully utilize your hardware capacity. You can write a program (called a "concurrent program") that runs as a concurrent process. Typically, you create concurrent programs for long-running, data- intensive tasks, such as posting a journal or generating a report.

PeopleSoft Integration Tools

Integration Broker

PeopleSoft Integration Broker is a middleware technology that facilitates synchronous and asynchronous messaging among internal systems and trading partners, while managing message structure, message format, and transport disparities.

PeopleSoft Integration Broker comprises two high-level subsystems: the Integration Engine and the Integration Gateway. The Integration Engine runs on the PeopleSoft application server. It is tied closely to PeopleSoft applications and produces or consumes messages for these applications.

The Integration Gateway is a platform that manages the actual receipt and delivery of messages passed among systems through the PeopleSoft Integration Broker. It provides support for the leading TCP/IP protocols used in the marketplace today, and more importantly, provides extensible interfaces for the development of new connectors for communication with legacy, ERP, and internet-based systems.

PeopleTools

PeopleTools refers to the collection of proprietary PeopleSoft tools for the development and execution of applications.

Component / Service Interface – APIs

A component interface is a set of application programming interfaces (APIs) that you can use to access and modify PeopleSoft database information using a program instead of the PeopleSoft client. PeopleSoft Component Interfaces expose a PeopleSoft component (a set of pages grouped for a business purpose) for synchronous access from another application (PeopleCode, Java, C/C++, or Component Object Model [COM]). A PeopleCode program or an external program (Java, C/C++, or COM) can view, enter, manipulate, and access PeopleSoft component data, business logic, and functionality without being online.

Application Message

Application Messages, or Message definitions, are the heart of PeopleSoft Integration Broker. They are templates for the data that the application sends or receives. Message definitions are created in PeopleSoft Application Designer. There are two types of PeopleSoft messages:

- Rowset-based (Structured) messages. For hierarchical data based on PeopleSoft records, these messages are created by assembling records, organizing them into a hierarchy, and selecting fields from those records to include in the message. The result is a rowset that doesn't need to match an existing rowset structure in the application. PeopleCode Rowset and Message classes are used to generate, send, receive, and process these messages.
- Non-rowset-based (Unstructured) messages. These messages can have virtually any structure and content. You create a message definition, but you do not insert any records. The message definition serves as a placeholder for the actual message. PeopleCode XmlDoc and Message classes are used to generate, send, receive and process these messages. You can also use the PeopleCode SoapDoc class to generate and process these messages for web services and SOAP messages.

File Layout

A File Layout object is a definitional map between a file (fixed format, comma separated values or XML) and a PeopleSoft Record definition. Creating File Layout objects requires just a few mouse clicks once you have a Record definition and sample file. Once you have created a File Layout object in PeopleTools 8.42 and higher, you can click the AE button in the Application Designer to have PeopleTools write the Application Engine PeopleCode for you that will load the file from the file server into PeopleSoft Records. For more information and examples see the Enterprise PeopleTools 8.46 PeopleBook: PeopleSoft Application Designer > Understanding File Layouts.

JD Edwards Web Services Gateway Components

Broker

The broker is a high-speed message router. It is the primary component of the platform's messaging facility. It uses a publish-and-subscribe model for asynchronous processing.

Integration Server

The integration server is the platform's central runtime component and the primary engine for the execution of the integration logic.

Integration Point

An Integration Point acts as the connection between the physical implementation and the logical business function. One can say that a system exposes its business functionality as Integration Points.

XBP

An XBP is a PeopleSoft marketing term, denoting specific Integration Points in the PeopleSoft EnterpriseOne product family applications that have been dubbed as such. Not all Integration Points exposed by PeopleSoft EnterpriseOne applications are XBP's.

Canonical

Canonical events are generic representations of data (both data structure and data content) created and published by a source integration within XPI Enterprise Integrator and subscribed to by one or more target integrations.

Cross-Application Integration Using BPEL

The table below provides a mapping of technology choices available in each product for a given integration pattern that can be orchestrated using BPEL PM.

Product	**Product/ Technology**	**Synchronous**	**Asynchronous**	**Batch**	**Message Payload**	**Supported Protocols / Required Tools**
Oracle E-Business	Business Event	NO	YES	NO	XML	Java
Oracle E-Business	Service Beans	YES	NO	NO	Service Data Objects	Java
Oracle E-Business	Web Service	YES	YES	NO	XML / Industry Standard/ SOAP	JMS, AQ, HTTP/ HTTPS
Oracle E-Business	XML Gateway	NO	YES	NO	XML / Industry Standard/ SOAP	JMS, AQ, HTTP/ HTTPS
Oracle E-Business	EDI Gateway	NO	YES	NO	EDI	File / FTP
Oracle E-Business	Open Interface / Concurrent Program	NO	NO	YES	Document Formats	PL/SQL / Database Connection
PeopleSoft	Component Interface	YES	NO	NO	Msg or SOAP	HTTP, HTTPS, SOAP
PeopleSoft	Application Messaging	YES	YES	YES	XML	Message Adapters
PeopleSoft	File Layout	NO	NO	YES	Flat File	File / FTP
JD Edwards	WSG Integration Point	YES	NO	NO	SOAP	Http/ Https
JD Edwards	XBP	YES	YES	NO	XML	

Orchestration	Product	Integration Points Synchronous	Integration Points Asynchronous
BPEL PM	Oracle eBusiness Suite	EBusiness Adapter to PL/SQL Service Beans	XML Gateway Business Events
BPEL PM	PeopleSoft	CI Web Service Application Messaging Web Service	Application Messaging Web Service
BPEL PM	JD Edwards	Integration Points (XBP Web Service)	NA

Service Discovery: Business Service Repository

Oracle Integration Repository

Oracle® Integration Repository is an integral part of Oracle E-Business Suite. It provides a complete catalog of Oracle E-Business Suite's Business Service interfaces. The tool aids users in easily discovering and deploying the appropriate Business Service interface for integration with any system, application, or business partner.

Oracle E-Business Suite 11i.10 components provide numerous programming interfaces that can be used by in-house development teams, SIs and ISVs to integrate E-Business applications with other applications in your environment.

Key Features of the Integration Repository

- Unified Repository from which all E-Business integration interface types are exposed
    - PL/SQL API
    - Java API
    - Concurrent Program
    - Open Interface
    - EDI Message
    - XML Message

  - Java Service Bean
  - Web Service
  - Business Event
  - Interface Tables
  - Views
  - Service Data Objects
- Repository updates are automated and documented
- Repository is the guaranteed single source of truth

The Repository is searchable on keywords and navigable either by interface type or product type.

Oracle Interactive Services Repository

The Interactive Services Repository (ISR) is a compilation of information about the numerous service endpoints exposed by the PeopleSoft Enterprise, JD Edwards OneWorld and JD Edwards World suites. PeopleSoft Integration Points, also known as EIPs or XPBs, are the web service connections that allow these applications to work smoothly with third-party systems and software, as well as other Oracle applications.

ISR works in conjunction with PeopleBooks to help users and implementation specialists fully explore the integration capabilities of PeopleSoft and JD Edwards applications.

Developers use ISR to document the technical structure of an integration point, its design patterns, and underlying technology. Analysts will use ISR to explore and understand integration points from a business process context. This is especially useful when implementing an Oracle-supported business process across multiple products and release versions.

Key Search Features of ISR

ISR contains high-level information about each integration point, including:

- Brief description
- Product family
- Product owner
- Business Process Association
- Data rules
- Setup rules

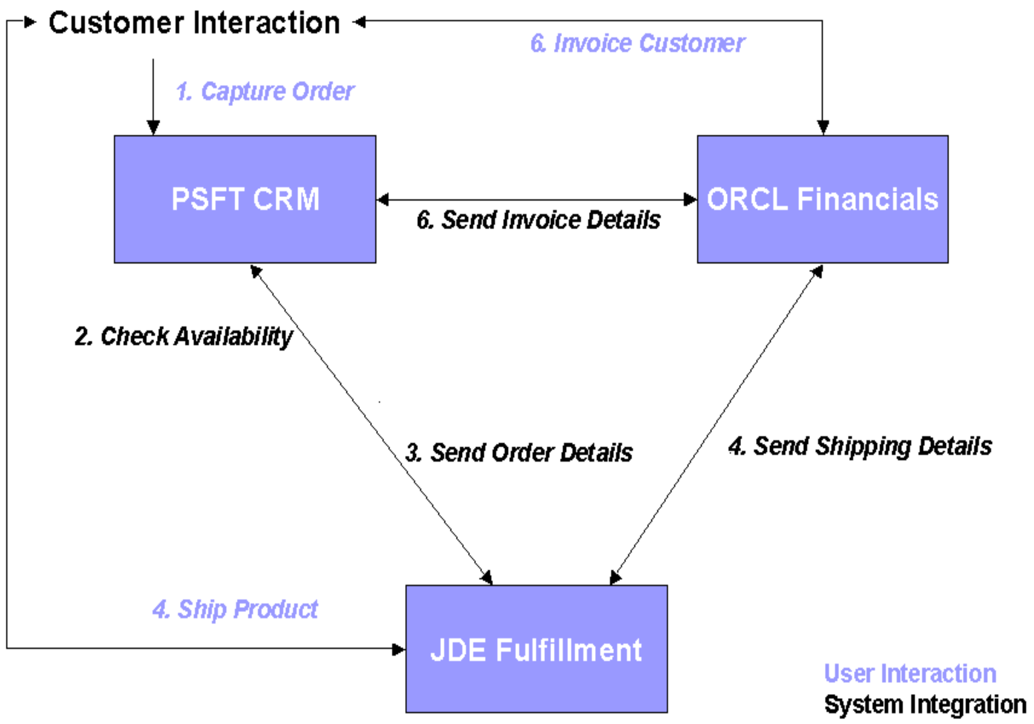
ISR enables you to quickly discover the integration points that are relevant to your specific needs. You can search for integration points by name, product, business process, and technology type. You can also discover point-to-point integrations between specific products.

CHAPTER 2: PROCESS ORIENTED INTEGRATION: A PRACTICAL USE CASE

Enterprise customers use more than one application to manage their business. With information residing in different applications it becomes difficult for customers to manage and plan their business activities. Integration overcomes the difficulties by enabling information stored in one application to be synchronized with information stored in another application.

This Use Case aims to integrate three different applications – Oracle E-Business Suite, PeopleSoft Enterprise and J.D. Edwards EnterpriseOne. "Order to Receipt" is one of the most common business processes that any company requires. As a result we have selected the *Order to Receipt* business flow to be the basis of this Use Case. The PeopleSoft CRM application will be used to capture the order; the JDE EnterpriseOne application will be used to fulfill (pick/ship) the order; and Oracle E-Business Suite will be used to Invoice the customer. BPEL PM will be the key integration platform to orchestrate the *Order to Receipt* business process between these three applications.

Below is the high level flow of the business process:



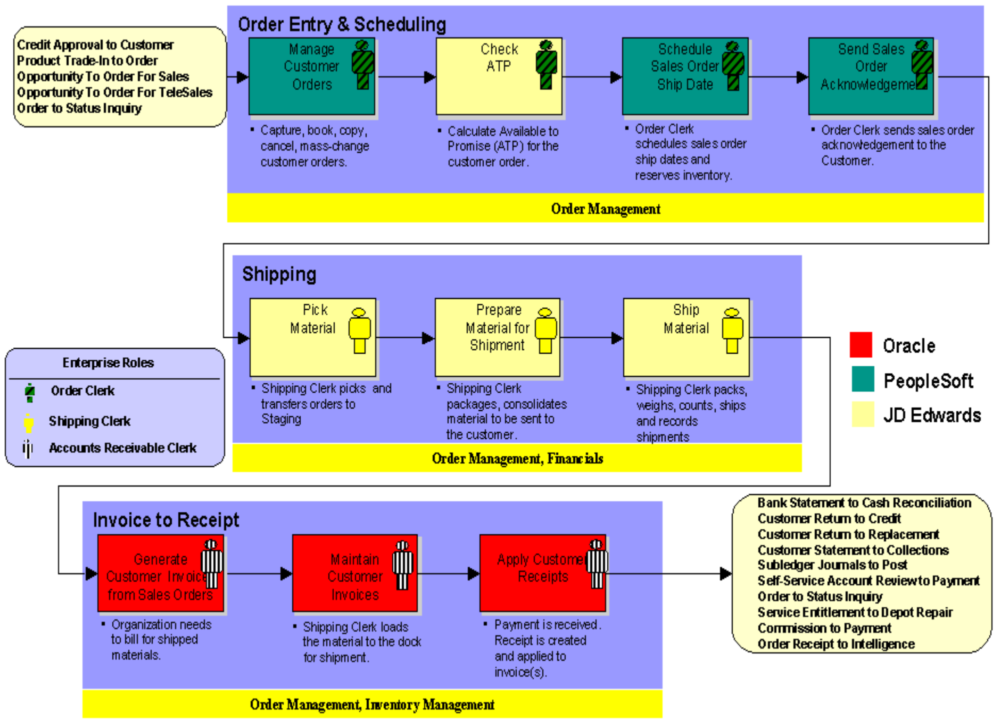
Use Case Assumptions

- Referential data will be manually created in all the three systems. Or, data from one of the systems will be used to update the other two systems. Referential data includes Company information, Branches, Customers, Items, etc.

- The business process will be kept as simple as possible. Workflows involving manual authorizations such as approvals for orders, approvals for shipment, etc., will be avoided.

Business Flow Breakdown

The *Order to Receipt* business process involves a number of activities. This use case assumes each *Order to Receipt* activity occurs in a specific application. Below is the break down of the activities and the applications where each occurs:


Credit Approval to Customer
Product Trade-In to Order
Opportunity To Order For Sales
Opportunity To Order For TeleSales
Order to Status Inquiry
Order Entry & Scheduling
Manage Customer Orders
Capture, book, copy, cancel, mass-change customer orders.
Check ATP
Calculate Available to Promise (ATP) for the customer order.
Schedule Sales Order Ship Date
Order Clerk schedules sales order ship dates and reserves inventory.
Send Sales Order Acknowledgeme
Order Clerk sends sales order acknowledgement to the Customer.
Order Management
Shipping
Pick Material
Shipping Clerk picks and transfers orders to Staging
Prepare Material for Shipment
Shipping Clerk packages, consolidates material to be sent to the customer.
Ship Material
Shipping Clerk packs, weighs, counts, ships and records shipments
Order Management, Financials
Enterprise Roles
Order Clerk
Shipping Clerk
Accounts Receivable Clerk
Oracle
PeopleSoft
JD Edwards
Invoice to Receipt
Generate Customer Invoice from Sales Orders
Organization needs to bill for shipped materials.
Maintain Customer Invoices
Shipping Clerk loads the material to the dock for shipment.
Apply Customer Receipts
Payment is received. Receipt is created and applied to invoice(s).
Order Management, Inventory Management
Bank Statement to Cash Reconciliation
Customer Return to Credit
Customer Return to Replacement
Customer Statement to Collections
Subledger Journals to Post
Self-Service Account Review to Payment
Order to Status Inquiry
Service Entitlement to Depot Repair
Commission to Payment
Order Receipt to Intelligence

Order to Receipt **Activity**	**Application**
Sales Order Entry	PeopleSoft Enterprise
Item Availability Check	J.D. Edwards EnterpriseOne
Pick, Pack and Ship	J.D. Edwards EnterpriseOne
Invoicing and A/R	Oracle E-Business Suite

Interface / Use Case 1: Item Availability

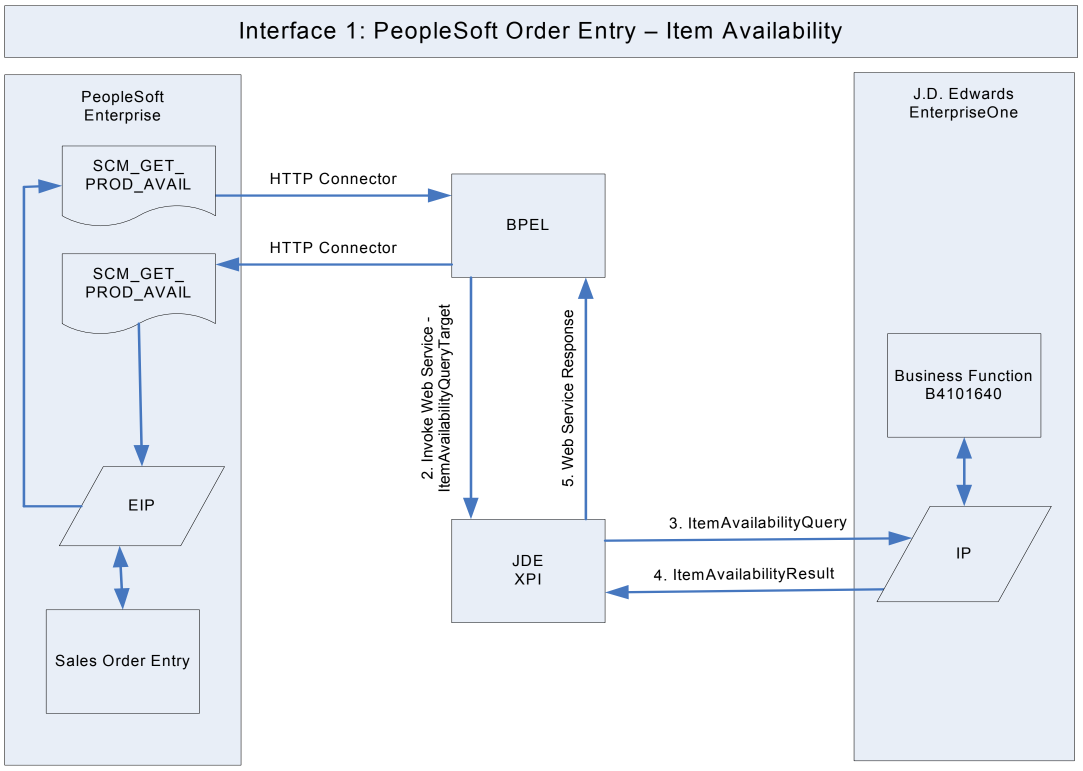
Source Application:	PeopleSoft Enterprise
Target Application:	J.D. Edwards EnterpriseOne
Process Type:	Synchronous
Products Involved:	PeopleSoft Enterprise, J.D. Edwards EnterpriseOne, J.D. Edwards XPI, Oracle BPEL PM

Note: The recommended practice is to look up the desired integration point in ISR.

http://www.peoplesoft.com/corp/en/products/dev_integration_portals/isr/index.jsp

Customer Relationship Management is handled in PeopleSoft Enterprise. PeopleSoft CRM is the primary source of customer information. All quotes and sales orders are captured in PeopleSoft CRM. Inventory Management on the other hand is handled in J.D. Edwards EnterpriseOne. Information such as Item Master details, Item Availability, etc., are stored in J.D. Edwards EnterpriseOne.

When an order is captured in PeopleSoft CRM it is necessary to check availability of the item in J.D. Edwards EnterpriseOne. This integration occurs in real time. It involves triggering a BPEL process that invokes a web service exposed by J.D. Edwards XPI. The web service executes a business function on the J.D. Edwards Enterprise Sever. The web service returns the quantity available in J.D. Edwards and PeopleSoft CRM should then decide if the order quantity can be fulfilled or not.


Interface 1: PeopleSoft Order Entry – Item Availability
PeopleSoft Enterprise
SCM_GET_ PROD_AVAIL
SCM_GET_ PROD_AVAIL
EIP
Sales Order Entry
HTTP Connector
HTTP Connector
BPEL
2. Invoke Web Service - ItemAvailabilityQueryTarget
5. Web Service Response
JDE XPI
3. ItemAvailabilityQuery
4. ItemAvailabilityResult
J.D. Edwards EnterpriseOne
Business Function B4101640
IP

Flow Details:

Step	Flow
1	Availability of an item is checked during order entry in PeopleSoft. A BPEL process is invoked that takes the request from PeopleSoft and sends it to JDE XPI.
2a	BPEL Process invokes ItemAvailabilityQueryTarget IP which is exposed by JDE as a web service.
2b	ItemAvailabilityQueryTarget IP to be exposed as a web service.
3	XPI EnterpriseOne adapter triggers the IP. EnterpriseOne processes the item availability request.
4	The EnterpriseOne Adapter receives the result and passes it on to the XPI Integration Server.
5	Result is passed to the BPEL PM .
6	BPEL PM receives the result, transforms it and sends it to PeopleSoft

Interface / Use Case 2: Order Confirmation

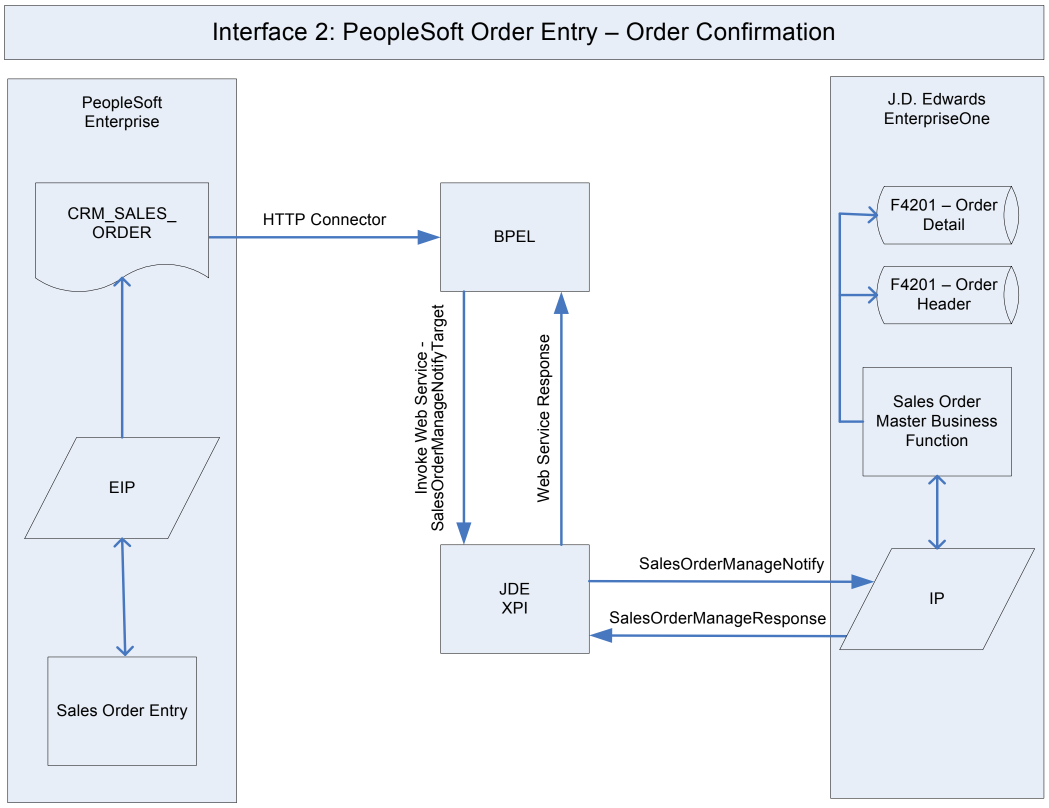
Source Application:	PeopleSoft Enterprise
Target Application:	J.D. Edwards EnterpriseOne
Process Type:	Asynchronous
Products Involved:	PeopleSoft Enterprise, J.D. Edwards EnterpriseOne, J.D. Edwards XPI, Oracle BPEL PM

Note: The recommended practice is to look up the desired integration point in ISR.

http://www.peoplesoft.com/corp/en/products/dev_integration_portals/isr/index.jsp

PeopleSoft CRM is the Order Management System and J.D. Edwards is the Fulfillment system. When a sales order is created in PeopleSoft CRM it is necessary to send the order details to J.D. Edwards EnterpriseOne to initiate fulfillment.

It involves triggering a BPEL process that invokes a web service exposed by J.D. Edwards XPI. The web service executes the Sales Order Master Business Function on the J.D. Edwards Enterprise Sever. The web service returns the result of the business function to the BPEL process that initiated it.



Flow Details:

Step	Flow
1	When the sales order is created in PeopleSoft a notification is sent to the BPEL PM.
2a	BPEL Process invokes SalesOrderManageNotifyTarget IP which is exposed by JDE as a web service.
2b	SalesOrderManageNotifyTarget IP to be exposed as a web service.
3	XPI EnterpriseOne adapter triggers the XBP. EnterpriseOne processes the Sales Order Master Business Function to generate the sales order in EnterpriseOne.
4	The EnterpriseOne Adapter receives the result and passes it on to the XPI Integration Server.
5	Result is passed to the BPEL PM.
6	BPEL PM receives the result and sends it to PeopleSoft.

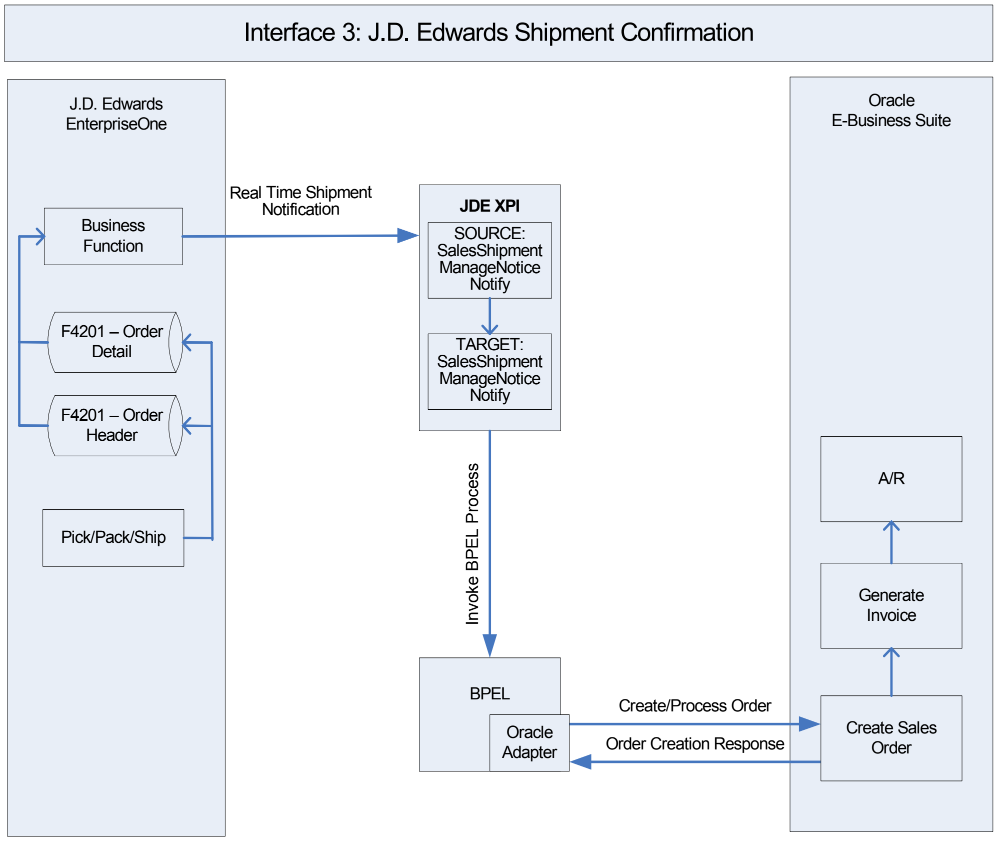
Interface / Use Case 3: Shipment Confirmation

Source Application:	J.D. Edwards EnterpriseOne
Target Application:	Oracle E-Business Suite
Process Type:	Asynchronous
Products Involved:	J.D. Edwards EnterpriseOne, Oracle E-Business Suite, J.D. Edwards XPI, Oracle BPEL PM

Note: The recommended practice is to look up the desired integration point in ISR for PeopleSoft / JD Ewards APIs and IREP for Oracle E-Business Suite APIs.

http://www.peoplesoft.com/corp/en/products/dev_integration_portals/isr/index.jsp

Since J.D. Edwards EnterpriseOne is the Order Fulfillment system, it handles the pick, pack and ship processes. The picking process is a standalone process within EnterpriseOne and no events are triggered. When the sales order is Ship Confirmed, a real time shipment notification is sent to the JDE Integration Server via the JDE EnterpriseOne adapter. In this POC, the shipment notification is subscribed by a BPEL process, which will then trigger a process in Oracle E-Business Suite.



Flow Details:

Step	Flow
1	When the sales order is shipped in J.D. Edwards EnterpriseOne, a real time notification is sent to the XPI EnterpriseOne Adapter.
2a	E1 processes the source message and converts it to a common view.
2b	A target IP will convert the common view to any desired format.
3	A BPEL process processes the target message and invokes the Oracle Application Adapter.
4	Oracle Application Adapter generates the order in E-Business Suite.
5	Result is passed to the BPEL PM.
6	BPEL PM receives the result and completes the process.

Note: Flow steps 3 to 6 are in one BPEL business process.

Interface / Use Case 4: Invoice Notification

Source Application:	Oracle E-Business Suite
Target Application:	J.D. Edwards EnterpriseOne
	PeopleSoft Enterprise
Process Type:	Asynchronous
Products Involved:	J.D. Edwards EnterpriseOne, Oracle E-Business Suite, PeopleSoft Enterprise, J.D. Edwards XPI, Oracle BPEL PM

Note: The recommended practice is to look up the desired integration point in ISR for PeopleSoft / JD Ewards APIs and IREP for Oracle E-Business Suite APIs.

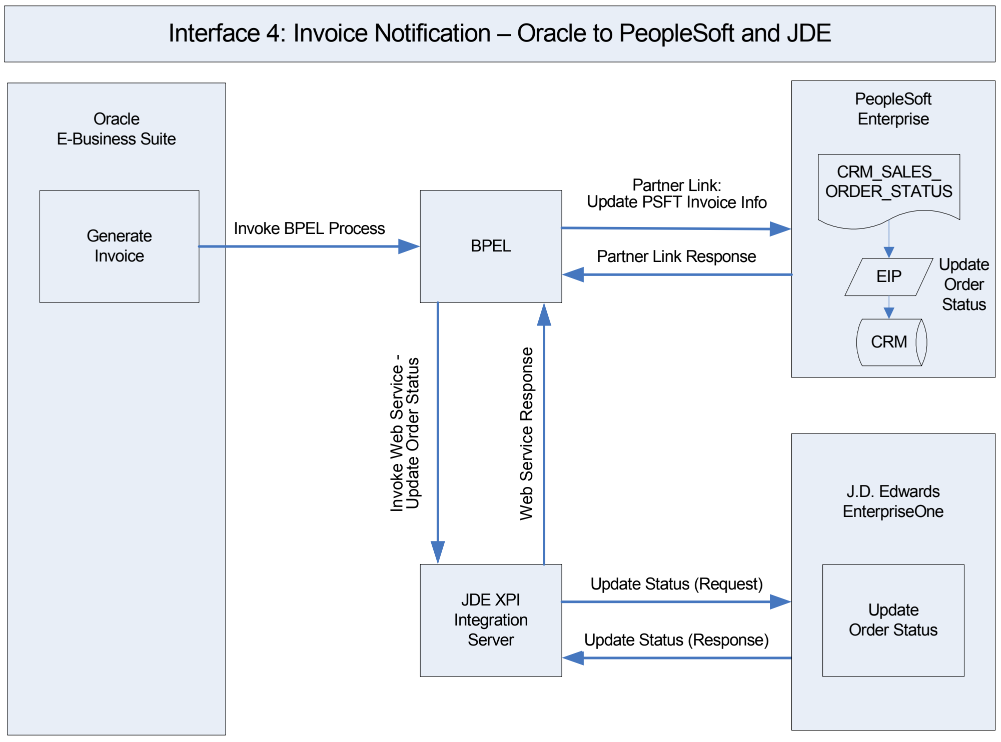
http://www.peoplesoft.com/corp/en/products/dev_integration_portals/isr/index.jsp

http://www.peoplesoft.com/corp/en/products/dev_integration_portals/isr/index.jsp

There are two events that will occur:

- Invoice Notification from Oracle E-Business Suite to PeopleSoft Enterprise
- Invoice Notification from Oracle E-Business Suite to J.D. Edwards EnterpriseOne

When the invoice is generated in Oracle E-Business Suite, the Oracle Application Adapter will trigger a notification and the BPEL process will be initiated. The BPEL process will have two partner links – one to invoke a PeopleSoft process and the other to invoke a J.D. Edwards web service.



Flow Details:

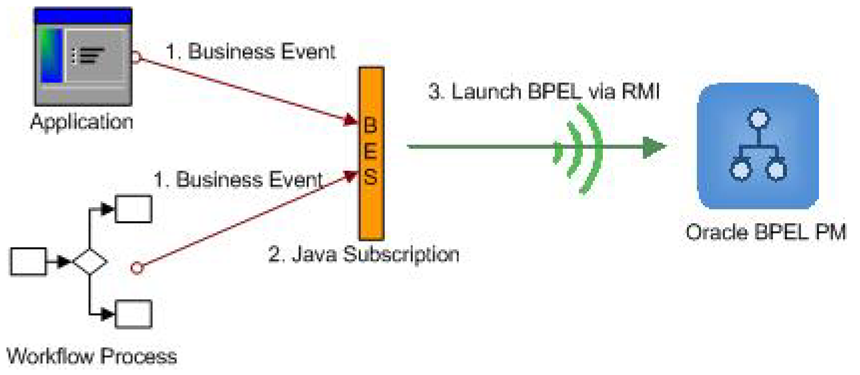
Step	Flow
1	When an invoice is generated in Oracle E-Business Suite, the notification is sent to the BPEL PM via the Oracle Application Adapter.
2	The BPEL process triggered initiates a partner link to PeopleSoft to update the invoice details.
3	PeopleSoft sends the response back to the BPEL PM.
4	The BPEL process also invokes a J.D. Edwards web service to update the invoice information in EnterpriseOne.
5	The "Order Status Update (?)" IP will be triggered to update the invoice information in EnterpriseOne.
6	The result from JDE E1 is passed to the BPEL PM.
7	BPEL PM receives the result and completes the process.

CHAPTER 3: HOW TO CONNECT BPEL AND ORACLE E-BUSINESS

BES to BPEL

The following sections document how a BPEL process can be launched from the Oracle E-Business Suite using the Business Event System.

Flow



The Concurrent Processing Tier must be upgraded to JDK 1.3 or the version of the Oracle E-Business Suite must be 11.5.8 or higher before performing this step.

The following is the simple 3-step procedure to launch a BPEL process from the Oracle E-Business Suite:

1. Applications within the Oracle E-Business Suite raise Business Events.
2. A Java Subscription for the business event raised triggers the BPEL process in the Oracle Application Server.
3. The BPEL Process is launched using the Java RMI interface in the Java Subscription.

Set up

The following set up steps are required before the BES – BPEL integration can be achieved.

Deploy Your BPEL Processes – Any BPEL process intended to be launched from the Oracle E-Business Suite must first be deployed in the Oracle BPEL Process Manager. Use the deployment tool, such as obant, provided with Oracle BPEL PM to deploy them.

Concurrent Processing Node Environment Settings – The Java deferred agent listeners are Java processes running on the Concurrent Processing node of your Oracle E-Business Suite instance. For the Java subscription to be executed correctly, you must copy the following jar files to the Concurrent Processing node and update the Java class path (AF_CLASSPATH) environment variable.

Note: Please use the standard procedures documented in the MetaLink Note 130091.1 to update the AF_CLASSPATH variable.

1. Obtain the following BPEL libraries and place them in a location, for example: /u01/app/apps11510/java/bpelbes/
   - orabpel-common.jar
   - orabpel-thirdparty.jar

Your Oracle E-Business Suite System Administrator must perform this step.

- orabpel.jar

2. Obtain the OC4J Client Side library, oc4jclient.zip and unzip it in a location, for example: /u01/app/apps11510/java/bpelbes/

You must have Workflow Administrator privileges to complete the Business Event System Registration

3. Locate the oc4jclient.jar file. This is the **only** jar file you need to include in the environment settings. Do not delete or change the directory structure created when the oc4jclient.zip is unzipped.

4. Create a Jar file for your Java Subscriptions, for example: /u01/app/apps11510/java/jbes/jar/mysubs.jar

You can also create new Business Events. However, you will need to raise the events within your application using the APIs provided by BES.

For the above example, the following is the AF_CLASSPATH setting:

/local/java/jdk1.4.2_04/lib/dt.jar:/local/java/jdk1.4.2_04/lib/tools.jar:/u01/app/apps11510/java/appsborg2.zip:/u01/app/apps11510/java/apps.zip:/slot06/appmgr/atg wfqa1ora/8.0.6/forms60/java:/u01/app/apps11510/java/u01/app/apps11510/java/bpelbes/oc4j/oc4jclient.jar:/u01/app/apps11510/java/bpelbes/orabpel-thirdparty.jar:/u01/app/apps11510/java/bpelbes/orabpel-common.jar:/u01/app/apps11510/java/bpelbes/orabpel.jar:/u01/app/apps11510/java/bpel-bes/mysubs.jar:

Deferred Agent Listeners

The Java Deferred Agent Listener must be running to process the asynchronous Java subscriptions.

Log on to the Oracle Workflow Manager to monitor and manage the listener processes.

Oracle JDeveloper Environment Settings

To test the invocation of BPEL processes within the Oracle JDeveloper environment, the following jar files must be added to the project libraries in addition to the BPEL and OC4J jar files mentioned above:

- fndctx.jar
- wfbes.jar
- wfjava.jar

You must append the jar files only at the end of the AF_CLASSPATH variable. You can find these jar files under $JAVA_TOP/oracle/apps/fnd/jar/ of your Oracle E-Business Suite instance.

Identify Business Event

You must first identify the business event that will be used to trigger a BPEL Process. Please refer to the "Managing Business Events" section of the *Oracle Workflow Developer's Guide* to understand how to search for business events.

Register Java Subscription

You must register a Local Java subscription to the event used to invoke the BPEL process. Please refer to the "Managing Business Events" section of the *Oracle Workflow Developer's Guide* to learn how to create subscriptions for business events. When defining a Java subscription, choose source type to be "Local", choose the "Custom" action type, and enter the Java class name in the "Java Rule Function" field.

The following table summarizes the subscription definition.

Source Type	**Phase**	**Action Type**	**Java Rule Function**
Local	>100	Custom	mypackage.myclass

RMI Interface

The Java class file in Appendix A, `InvokeBPELSub.java` provides you with a sample Java subscription. The subscription class shows how the Java RMI interface provided by Oracle BPEL Process Manager is used to invoke the CreditRatingService process.

Message Structure

The Java subscription is responsible for constructing the message structure required by the BPEL process being launched. In the sample subscription, the input message to the CreditRatingService process is constructed within the Java Subscription by obtaining the SSN value from the event, and creating the XML string. You could also use the generate function capability of BES to generate the message when the event is raised. Please refer to BES documentation for using the generate function. However, please note that BES does not perform any XML content verification of the generated message before processing the Java subscription.

Security

The subscription sample hard codes the BPEL server information and any authentication required when submitting the request. You must store the authentication information in your own schema/tables in a secure manner. Tools such as the DBMS_OBFUSCATE database package can be leveraged to encrypt and decrypt password values.

Limitations

This technical white paper is designed to work only with deferred Java subscriptions (phase >100). Synchronous Java subscriptions for events raised in the Apache middle tier are not supported.

Known Issues

The version of the DeliveryService class that is used for compiling the Java Subscription must match the version of the class in the BPEL runtime environment. If this does not match, you may get the following error message:

Raising Business Events

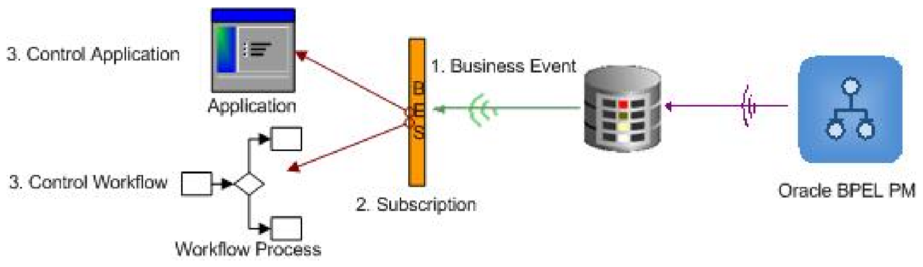
Applications can leverage BES to raise events in both PL/SQL and Java. When events are raised in PL/SQL, the invocation of BPEL processes is always asynchronous and handled by the Java deferred agent listeners. When events are raised in Java, the invocation of BPEL processes can be synchronous. However, the scope of this white paper is limited to asynchronous subscriptions (phase > 100) only.

The Java class file in Appendix B, RaiseEvent.java provides you with a sample Java class which shows how to raise a business event in Java. You can also choose to raise these events within a workflow process using the "Raise" event activity.

BPEL TO BES

The following sections document how a BPEL process can communicate asynchronously with Oracle E-Business Suite using BES.

Flow



The following is the simple 3-step procedure to trigger business logic in Oracle E-Business Suite from BPEL

1. The database adapter shipped with the Oracle BPEL Process Manager is used to connect to the Oracle E-Business Suite.
2. The BPEL Process Manager raises a Business Event within an invoke activity, linked to the database adapter service
3. The Event Manager within the Oracle E-Business Suite will then process subscriptions to this event. The subscription can trigger application logic or a workflow process.

Set up

Register Business Event – Register the business event to the process interactions from BPEL. Please refer to the "Managing Business Events" section of the *Oracle Workflow Developer's Guide* to learn how to create business events. If you already have an event, you can skip this step and proceed to the next step.

Register Subscription – You must register a "Local" subscription to the event raised by the BPEL process. Please refer to the "Managing Business Events" section of the *Oracle Workflow Developer's Guide* to learn how to create subscriptions for business events. You can leverage the entire range of capabilities of BES when processing events raised by BPEL. You can choose any Action Type as per your needs. You can send the event to a waiting workflow, execute custom PL/SQL logic, or execute Java logic. When defining a Java subscription, choose source type to be "Local", choose the "Custom" action type, and enter the Java class name in the "Java Rule Function" field.

Source Type	Phase	Action Type
Local	>100	Any

BPEL invoke Activity

Database Adapter Service

> The database adapter service of the Oracle BPEL Process Manager is used to connect to the Oracle E-Business Suite instance and use the PL/SQL interface provided by BES to raise the Business Event. You must create a new PartnerLink that uses the Database Adapter Service. When creating the PartnerLink, choose "Call a Stored Procedure" as the operation type, select the "Apps" schema, and enter "WF_EVENT.RAISE" as the stored procedure name.

Invoke Activity

> The invoke activity in your BPEL process must be linked to the PartnerLink created in the previous section. The EVENT_DATA must contain the XML string that the subscription to the Event being raised (EVENT_NAME) expects.

Parameters and Message Structure

> The BPEL process designer is responsible for constructing the message structure as required by the subscription before the event is raised. The EVENT_DATA parameter must be mapped to the message structure.
>
> The EVENT_NAME parameter must contain the event name being raised.
>
> The EVENT_KEY parameter must contain the event key.

BPEL to Oracle E-Business API

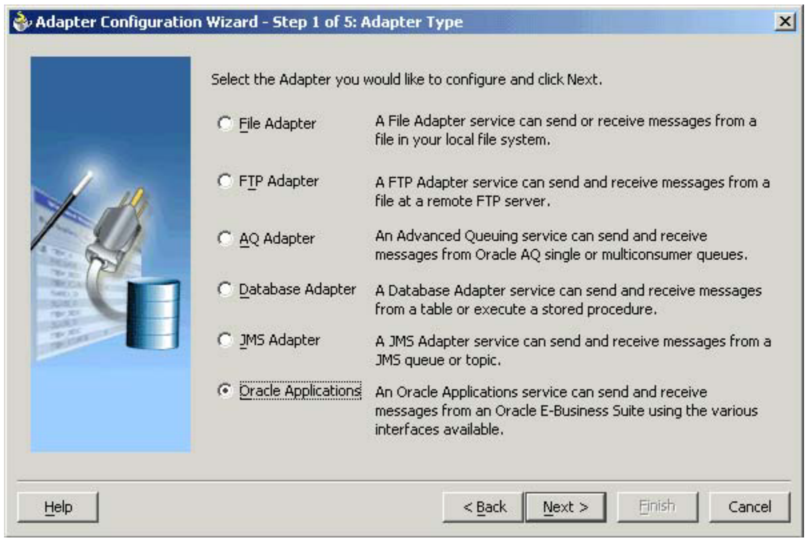
The Oracle AS Adapter for Oracle Applications is part of the BPEL PM 10.1.2 install and is available for both standalone as well as middle-tier installations. The Oracle AS Adapter for Oracle Applications is a pure JCA 1.5 Resource Adapter and is deployed in the OC4J container in managed mode. The Adapter SDK, Framework and Toolkit are one and the same and are used for the bidirectional integration of the JCA 1.5 resource adapters with BPEL Process Manager. Adapter SDK is based on open standards and employs the Web Service Invocation Framework (WSIF) technology for exposing the underlying JCA Interactions as Web Services.

Configuring the Oracle Applications Create Customer Service

Note: The recommended practice is to first discover the desired API in Integration Repository for version 11.5.10 available at URL. This guarantees backwards comptability because all the APIs in Integration Repository are public APIs. Use the adapter to bind to the desired API.

The Jdeveloper-based design-time wizard is used for configuration of the Oracle AS Adapters and is launched from the BPEL Partner Link activity. WSDL files are created for both JCA Outbound (Request-Response service – BPEL invoke) and JCA Inbound (Event Notification – BPEL receive) Interactions. The following set of figures illustrates the configuration of the create customer service using the Jdeveloper tool.

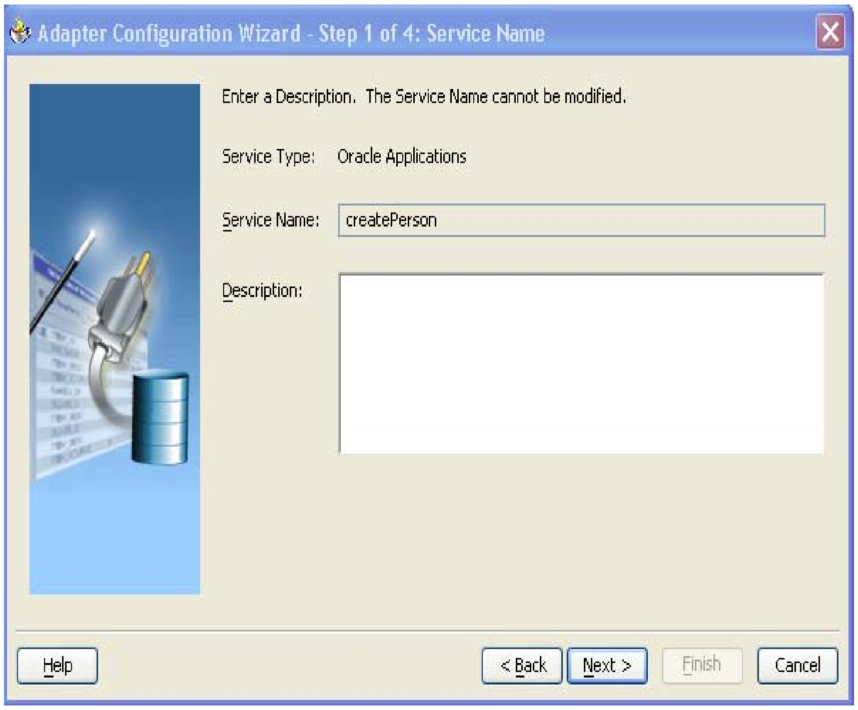
Pick **Oracle Applications** from the list of Adapters.


Adapter Configuration Wizard - Step 1 of 5: Adapter Type

Select the Adapter you would like to configure and click Next.

○ File Adapter	A File Adapter service can send or receive messages from a file in your local file system.
○ FTP Adapter	A FTP Adapter service can send and receive messages from a file at a remote FTP server.
○ AQ Adapter	An Advanced Queuing service can send and receive messages from Oracle AQ single or multiconsumer queues.
○ Database Adapter	A Database Adapter service can send and receive messages from a table or execute a stored procedure.
○ JMS Adapter	A JMS Adapter service can send and receive messages from a JMS queue or topic.
● Oracle Applications	An Oracle Applications service can send and receive messages from an Oracle E-Business Suite using the various interfaces available.

In the **Service Name** field, enter a service name.


Adapter Configuration Wizard - Step 1 of 4: Service Name

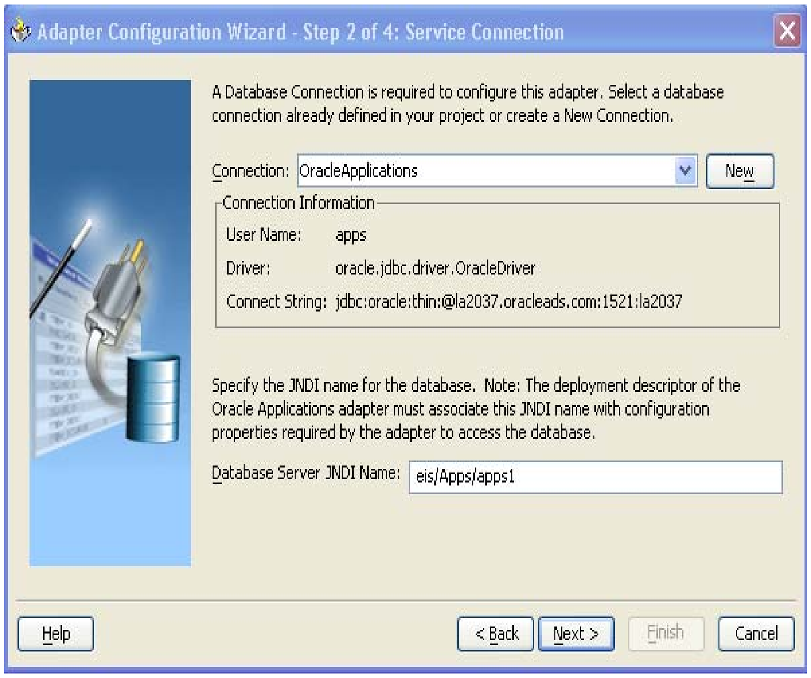
Enter a Description. The Service Name cannot be modified.

Service Type: Oracle Applications

Service Name: createPerson

Description:

Specify the connection parameters as shown in the below figure. The JNDI name refers to the runtime connection and points to the logical deployment of the adapter**.**


Adapter Configuration Wizard - Step 2 of 4: Service Connection

A Database Connection is required to configure this adapter. Select a database connection already defined in your project or create a New Connection.

Connection: OracleApplications

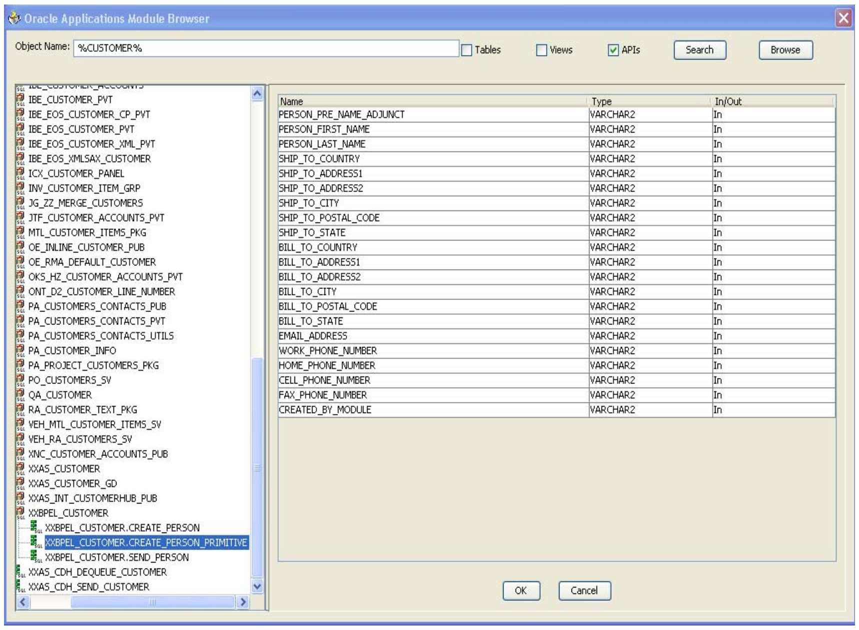
Connection Information

User Name:	apps
Driver:	oracle.jdbc.driver.OracleDriver
Connect String:	jdbc:oracle:thin:@la2037.oracleads.com:1521:la2037

Specify the JNDI name for the database. Note: The deployment descriptor of the Oracle Applications adapter must associate this JNDI name with configuration properties required by the adapter to access the database.

Database Server JNDI Name: eis/Apps/apps1

Use the **browse** option to select the Customer record. The syntax for browsing is based on SQL 92 standard.

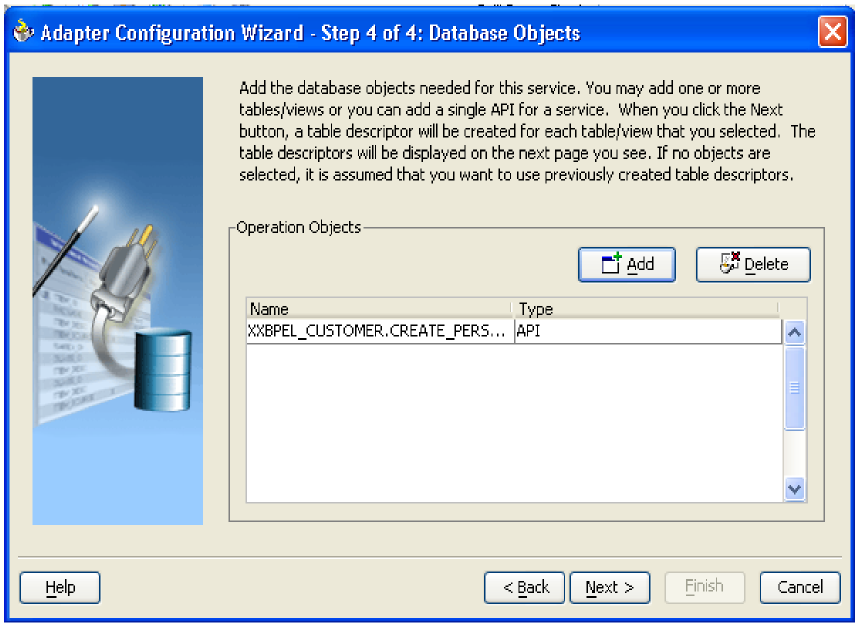


Object Name: %CUSTOMER%

Tables [ ] Views [ ] APIs [✓]

```
IBE_CUSTOMER_PVT
IBE_EOS_CUSTOMER_CP_PVT
IBE_EOS_CUSTOMER_PVT
IBE_EOS_CUSTOMER_XML_PVT
IBE_EOS_XMLSAX_CUSTOMER
ICX_CUSTOMER_PANEL
INV_CUSTOMER_ITEM_GRP
JG_ZZ_MERGE_CUSTOMERS
JTF_CUSTOMER_ACCOUNTS_PVT
MTL_CUSTOMER_ITEMS_PKG
OE_INLINE_CUSTOMER_PUB
OE_RMA_DEFAULT_CUSTOMER
OKS_HZ_CUSTOMER_ACCOUNTS_PVT
ONT_D2_CUSTOMER_LINE_NUMBER
PA_CUSTOMERS_CONTACTS_PUB
PA_CUSTOMERS_CONTACTS_PVT
PA_CUSTOMERS_CONTACTS_UTILS
PA_CUSTOMER_INFO
PA_PROJECT_CUSTOMERS_PKG
PO_CUSTOMERS_SV
QA_CUSTOMER
RA_CUSTOMER_TEXT_PKG
VEH_MTL_CUSTOMER_ITEMS_SV
VEH_RA_CUSTOMERS_SV
XNC_CUSTOMER_ACCOUNTS_PUB
XXAS_CUSTOMER
XXAS_CUSTOMER_GD
XXAS_INT_CUSTOMERHUB_PUB
XXBPEL_CUSTOMER
    XXBPEL_CUSTOMER.CREATE_PERSON
    XXBPEL_CUSTOMER.CREATE_PERSON_PRIMITIVE
    XXBPEL_CUSTOMER.SEND_PERSON
XXAS_CDH_DEQUEUE_CUSTOMER
XXAS_CDH_SEND_CUSTOMER
```

Name	Type	In/Out
PERSON_PRE_NAME_ADJUNCT	VARCHAR2	In
PERSON_FIRST_NAME	VARCHAR2	In
PERSON_LAST_NAME	VARCHAR2	In
SHIP_TO_COUNTRY	VARCHAR2	In
SHIP_TO_ADDRESS1	VARCHAR2	In
SHIP_TO_ADDRESS2	VARCHAR2	In
SHIP_TO_CITY	VARCHAR2	In
SHIP_TO_POSTAL_CODE	VARCHAR2	In
SHIP_TO_STATE	VARCHAR2	In
BILL_TO_COUNTRY	VARCHAR2	In
BILL_TO_ADDRESS1	VARCHAR2	In
BILL_TO_ADDRESS2	VARCHAR2	In
BILL_TO_CITY	VARCHAR2	In
BILL_TO_POSTAL_CODE	VARCHAR2	In
BILL_TO_STATE	VARCHAR2	In
EMAIL_ADDRESS	VARCHAR2	In
WORK_PHONE_NUMBER	VARCHAR2	In
HOME_PHONE_NUMBER	VARCHAR2	In
CELL_PHONE_NUMBER	VARCHAR2	In
FAX_PHONE_NUMBER	VARCHAR2	In
CREATED_BY_MODULE	VARCHAR2	In


Adapter Configuration Wizard - Step 4 of 4: Database Objects

Add the database objects needed for this service. You may add one or more tables/views or you can add a single API for a service. When you click the Next button, a table descriptor will be created for each table/view that you selected. The table descriptors will be displayed on the next page you see. If no objects are selected, it is assumed that you want to use previously created table descriptors.

Operation Objects

Name	Type
XXBPEL_CUSTOMER.CREATE_PERS...	API

Choose the **XXBPEL_CUSTOMER_CREATE_PERSON_PRIMITIVE** API. This contains only primitive data types. The current version of the Oracle AS Adapter for Oracle Applications does not support PL/SQL boolean and PL/SQL Records. You need to implement wrapper code for the above type support. The Adapter Service created can be deployed and invoked from a BPEL process.

CHAPTER 4: HOW TO CONNECT BPEL AND PEOPLESOFT APPLICATION

BPEL Consuming Peoplesoft Web Service

The following section briefly outlines how to connect BPEL with a PeopleSoft Component Interface.

1. Generate the WSDL for a given Component Interface using Peopletools and save the definition in the File system.
2. Create a BPEL Process using the designer and bind the partner link to the WSDL generated through Peopletools.
3. Complete the BPEL Process specification compile and run the process.

PeopleSoft Providing Synchronous Web Services Using Component Interfaces (SOAPTOCI)

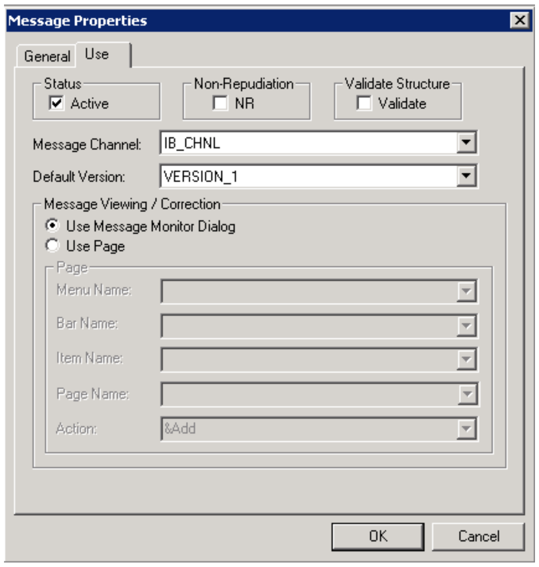
Component Interface (CI) based web services reuse existing business logic, do not require any additional coding, leverage user and row-level application security, and are easy to set up and maintain. For components with audit fields, the "last updated by" field will always show the user ID that was passed in the SOAP or HTTP header to invoke the CI.

The following steps are one-time setup steps for each PeopleSoft environment that will be providing CI-based synchronous web services:

- Verify that a default server is specified in the integrationGateway.Properties file on the PeopleSoft web server.
- Each PeopleSoft system that will provide CI-based web services must have its own Integration Broker and must have a default server specified.

  Sample entry in integrationGateway.properties file:
  ```
  ig.isc.serverURL=//<app server>:<jolt port number>
  ig.isc.userid=<ID that starts app server - not VP1 or super
  user>
  ig.isc.password=<encrypted with PSCIPHER.bat or PSCIPHER.sh>
  ig.isc.toolsRel=<PeopleTools release you see when you type CNTL
  +J in PIA>
  ```

- Open SOAPTOCI message in the Application Designer and make sure the Active checkbox is checked in the Properties page:


Message Properties

General | Use

Status: [✓] Active

Non-Repudiation: [ ] NR

Validate Structure: [ ] Validate

Message Channel: IB_CHNL

Default Version: VERSION_1

Message Viewing / Correction: [● Use Message Monitor Dialog] ○ Use Page

Page

Menu Name:

Bar Name:

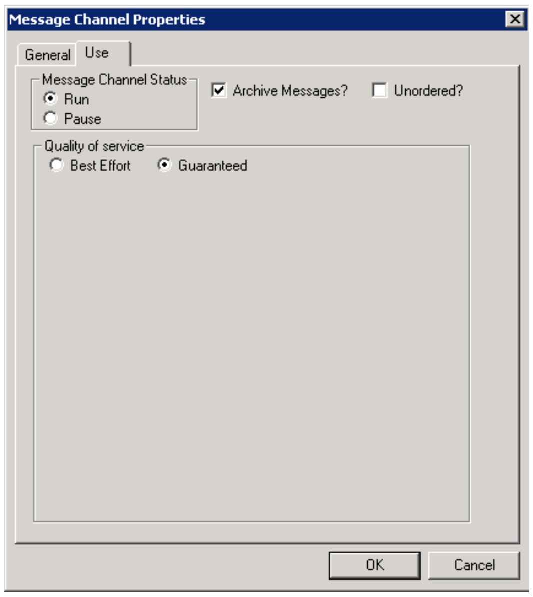
Item Name:

Page Name:

Action: &Add

OK | Cancel

In PIA or in the Application Designer, make sure the IB_CHNL channel is running and that you and/or the administrator have the required permissions to see the channel in the Message Monitor.


Message Channel Properties

General | Use

Message Channel Status: [● Run] ○ Pause

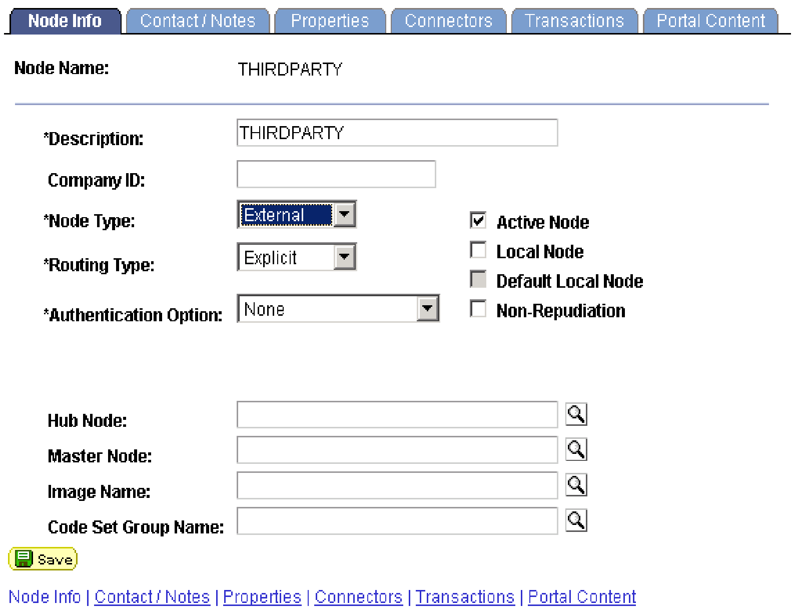
[✓] Archive Messages?

[ ] Unordered?

Quality of service: ○ Best Effort [● Guaranteed]

OK | Cancel

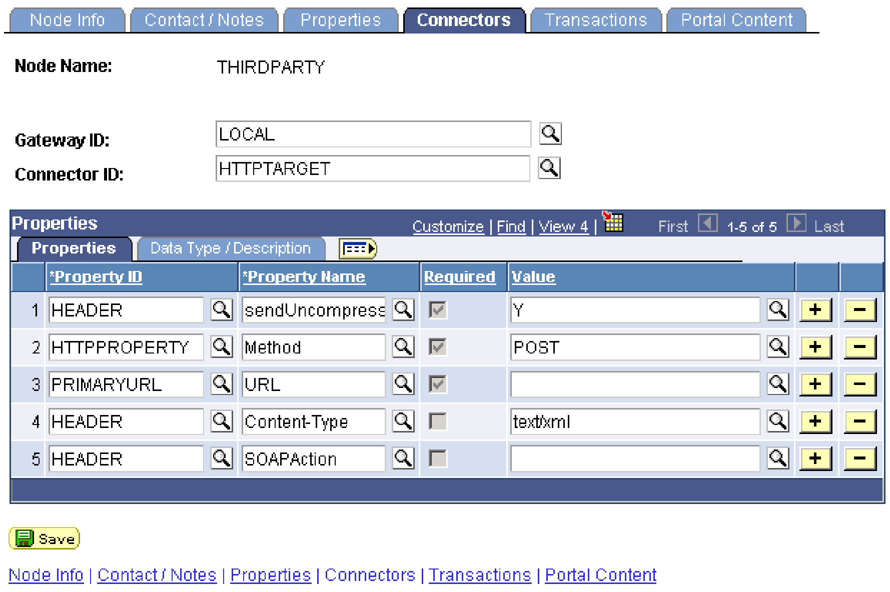
Create a new Message Node in PIA that will represent all external systems that will invoke the CI-based web services. The name of this node is arbitrary. Choose one that makes sense for your environment. In the Node Info tab, specify Node Type: External, Routing Type: Explicit, and set the node to Active. For Authentication, either choose “None” or “SSL,” depending on your corporate security policy. In the example below, the node is called “THIRDPARTY.”


Node Info | Contact / Notes | Properties | Connectors | Transactions | Portal Content
Save

Node Name:	THIRDPARTY
*Description:	THIRDPARTY
Company ID:	
*Node Type:	External
*Routing Type:	Explicit
*Authentication Option:	None
Hub Node:	
Master Node:	
Image Name:	
Code Set Group Name:	

- [✓] Active Node
- [ ] Local Node
- [ ] Default Local Node
- [ ] Non-Repudiation

Set the following properties in the Connectors tab:

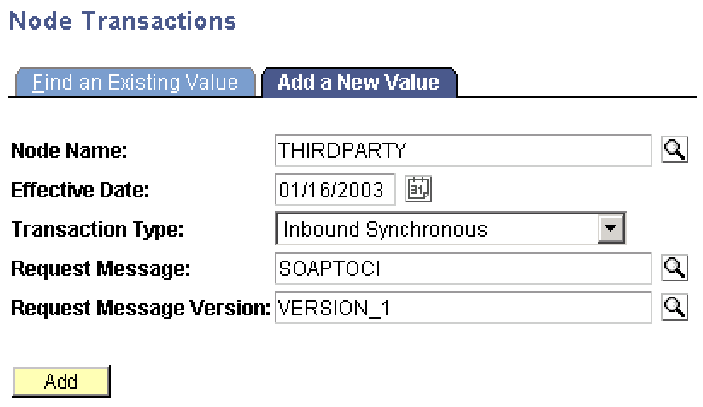

Node Info | Contact / Notes | Properties | Connectors | Transactions | Portal Content
Customize | Find | View 4 | First 1-5 of 5 Last
Properties | Data Type / Description
Save

Node Name:	THIRDPARTY
Gateway ID:	LOCAL
Connector ID:	HTTPTARGET

Properties

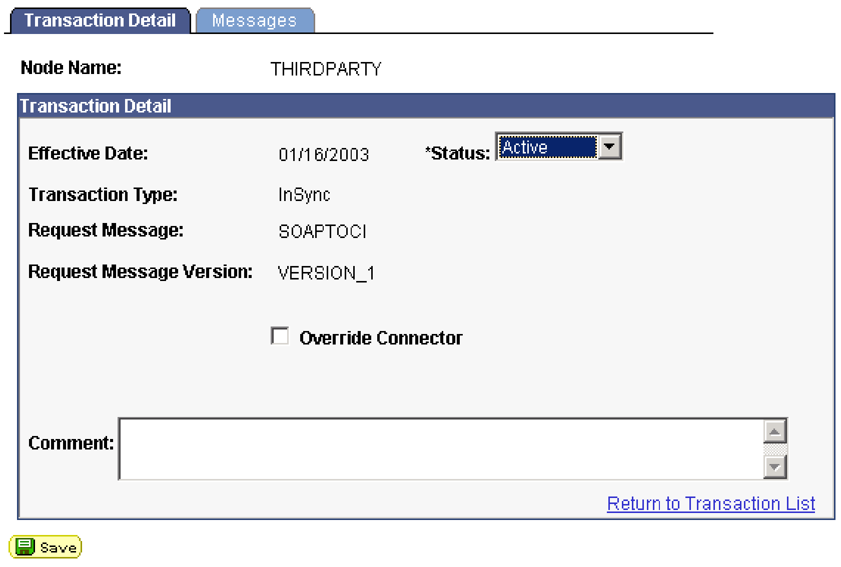
	*Property ID	*Property Name	Required	Value
1	HEADER	sendUncompress	✓	Y
2	HTTPPROPERTY	Method	✓	POST
3	PRIMARYURL	URL	✓	
4	HEADER	Content-Type		text/xml
5	HEADER	SOAPAction		

On the Transactions tab, add the SOAPTOCI message as both inbound synchronous and outbound synchronous transactions for the node.



Node Transactions

Node Name:	THIRDPARTY
Effective Date:	01/16/2003
Transaction Type:	Inbound Synchronous
Request Message:	SOAPTOCI
Request Message Version:	VERSION_1



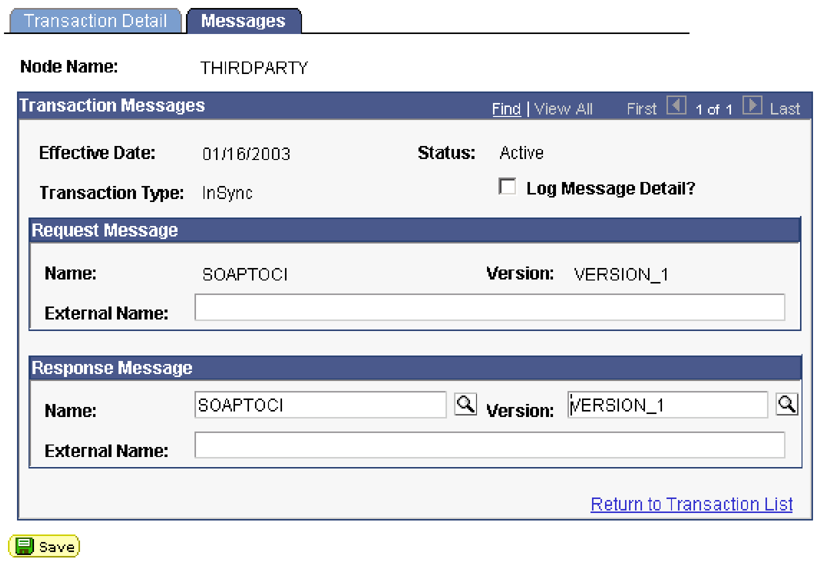
Node Name: THIRDPARTY

Transaction Detail

Effective Date:	01/16/2003
*Status:	Active
Transaction Type:	InSync
Request Message:	SOAPTOCI
Request Message Version:	VERSION_1

☐ Override Connector

Comment:


Transaction Detail | Messages

Node Name: THIRDPARTY

Transaction Messages Find | View All First 1 of 1 Last

Effective Date: 01/16/2003 Status: Active

Transaction Type: InSync ☐ Log Message Detail?

Request Message

Name: SOAPTOCI Version: VERSION_1

External Name:

Response Message

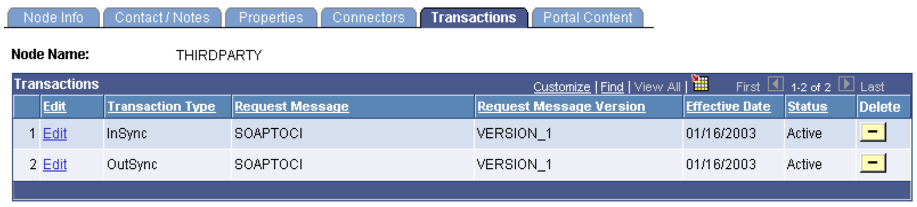
Name: SOAPTOCI Version: VERSION_1

External Name:

Return to Transaction List

Save

When you have finished, you will have the following two transactions on the node:


Node Info | Contact/Notes | Properties | Connectors | Transactions | Portal Content

Node Name: THIRDPARTY

Transactions Customize | Find | View All First 1-2 of 2 Last

	Edit	Transaction Type	Request Message	Request Message Version	Effective Date	Status	Delete
1	Edit	InSync	SOAPTOCI	VERSION_1	01/16/2003	Active	−
2	Edit	OutSync	SOAPTOCI	VERSION_1	01/16/2003	Active	−

The following steps must be performed for each CI that you want to web service enable:

- In the Application Designer, either verify the existence of, or create and test a component interface based on the component you want to web service enable.
  For more information on creating and testing CI's, see the Enterprise PeopleTools 8.46 PeopleBook: PeopleSoft Component Interfaces
  http://peoplebooks.peoplesoft.com:8900/PSOL/pt846/eng/psbooks/index.htm
- Create a permission list and role for the CI. The permission list and role should not have any menu navigation privileges. If someone were to try to log in to the application in a web browser or Application Designer using an ID with this role/permission list, they should be denied login or not be able to access anything.
- Grant the permission list you created in the above step access to the SOAPTOCI web library.
- Grant the role you created above to the appropriate PeopleSoft user ID's. If the user has only the above role/permission list, then the user will not be able to do anything in the application other than execute the named component interface.
- In the Integration Broker Web Services Menu, select the CI and click the WSDL link to generate the WSDL that BPEL PM will consume. You must have permissions on each method of the CI that you want to generate WSDL for.

- Save the URL so that BPEL PM will be able to consume the WSDL as a PartnerLink

PeopleSoft Providing Message Based Synchronous Web Services

Synchronous messages can be exposed via web services, regardless if they are rowset-based (structured) or nonrowset-based (unstructured). This section explains how to web service enable Application Messages as synchronous web services.

Note: If an unstructured message does not have an associated schema, the schema section of the WSDL document will be incomplete. You must either associate a schema with the message or modify the schema section of the WSDL so that the WSDL document completely describes the message. If you are using a delivered unstructured message where a schema has been associated, that schema will appear in the WSDL.

This functionality uses the basic PeopleSoft Integration Broker infrastructure with one exception: At runtime SOAP request messages are sent to a specific listening connector on the gateway.

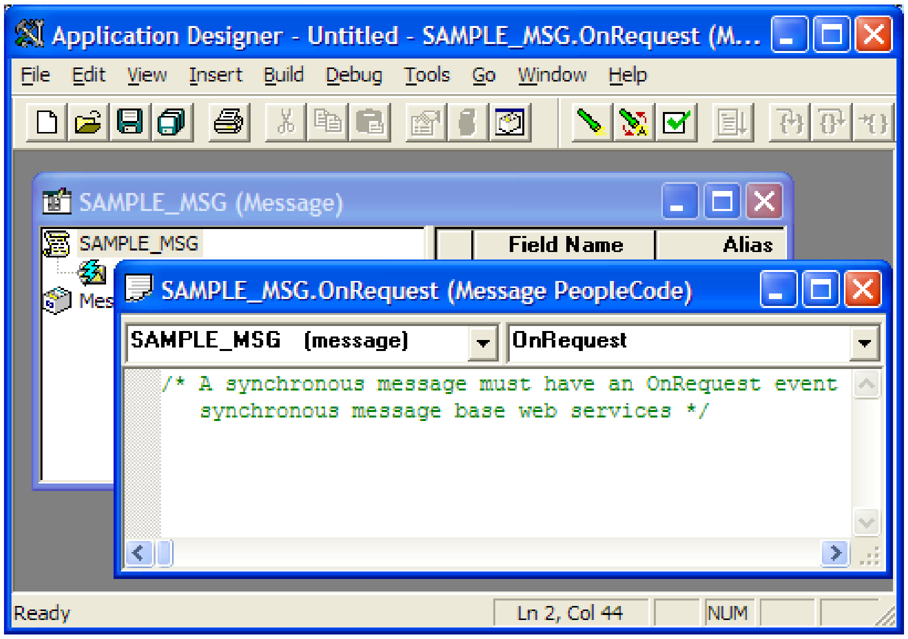
External parties send web service requests to the PeopleSoftServiceListeningConnector, which is responsible for stripping the SOAP envelope and forwarding the request to the application server for processing. The application server receives the request as it would any other message, and invokes the appropriate message processing.

There are two primary differences between CI and message based web services:

- In a CI based web service, all of the logic already exists in the component. You do not have to add any business logic to a CI. In Application message based web services, if there is no delivered logic in a message event, you must code the logic yourself.
- To invoke a CI based web service, the invoking system must always supply a valid PeopleSoft username and password, with the correct permissions, in order to invoke the specified service.

  Application Message based web services never take a username and password. They can be secured, however, using digital certificates, SSL, or a node password. For more information see Enterprise PeopleTools 8.46 PeopleBook: Integration Broker > Setting Up Secure Messaging Environments.

For synchronous message based web services, the request message must have an OnRequest event. See below:

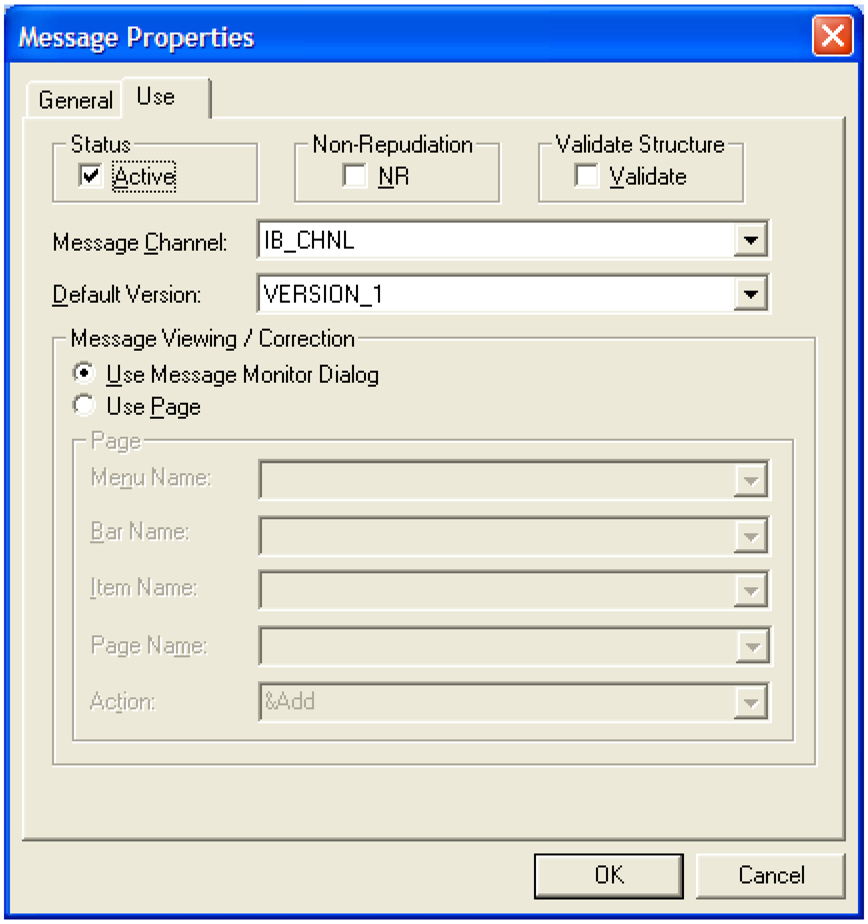

Application Designer - Untitled - SAMPLE_MSG.OnRequest (M...
File Edit View Insert Build Debug Tools Go Window Help
SAMPLE_MSG (Message)
SAMPLE_MSG
Mes
Field Name
Alias
SAMPLE_MSG.OnRequest (Message PeopleCode)
SAMPLE_MSG (message)
OnRequest
Ready
Ln 2, Col 44
NUM

```
/* A synchronous message must have an OnRequest event
   synchronous message base web services */
```

The OnRequest event generally contains the code to send back to the requestor a response message. The response message is generally a separate message that does not need to contain any code.

Web Service Enable Synchronous Messages:

In the Application Designer, make sure the request and response messages are Active. The Message Channel does not matter as synchronous messages do not get queued in channels. If you are creating a news message and need to assign a channel, either create a channel just for synchronous messages or use the delivered IB_CHNL.



Status: [✓ Active]

Non-Repudiation: [ ] NR

Validate Structure: [ ] Validate

Message Channel: IB_CHNL

Default Version: VERSION_1

Message Viewing / Correction: [● Use Message Monitor Dialog] Use Page

Page

Menu Name:

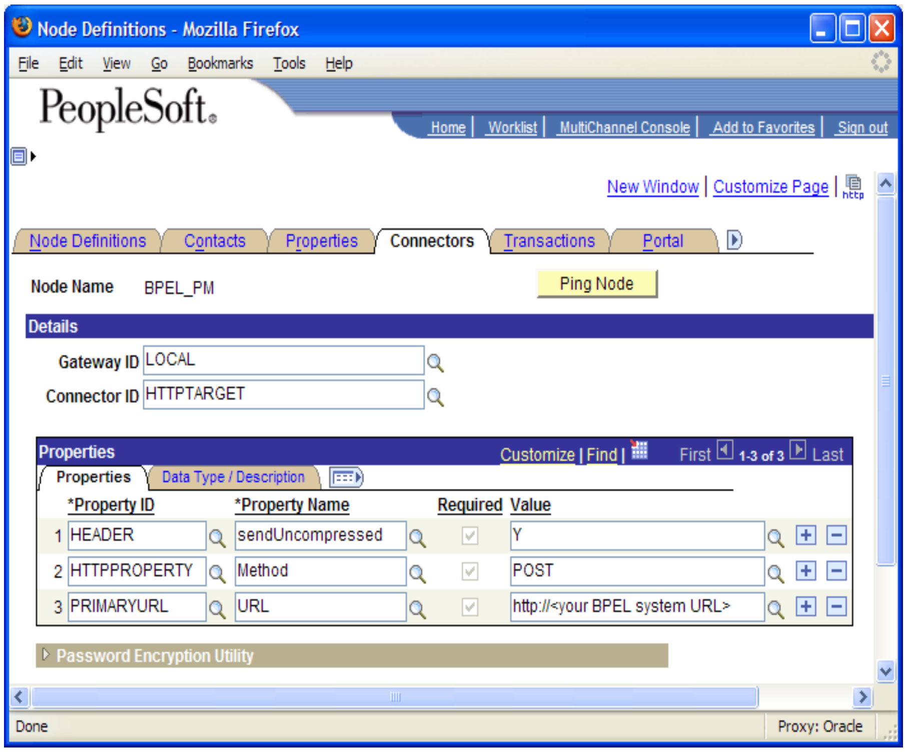
Bar Name:

Item Name:

Page Name:

Action: &Add

In the Integration Broker Node Definitions, either verify the existence of or create a node definition for the BPEL PM system.


Node Definitions - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
PeopleSoft.
Home | Worklist | MultiChannel Console | Add to Favorites | Sign out
New Window | Customize Page |
Node Definitions Contacts Properties Connectors Transactions Portal
Ping Node
Customize | Find | First 1-3 of 3 Last
Properties Data Type / Description
Password Encryption Utility
Done
Proxy: Oracle

Node Name BPEL_PM

Details

Gateway ID LOCAL

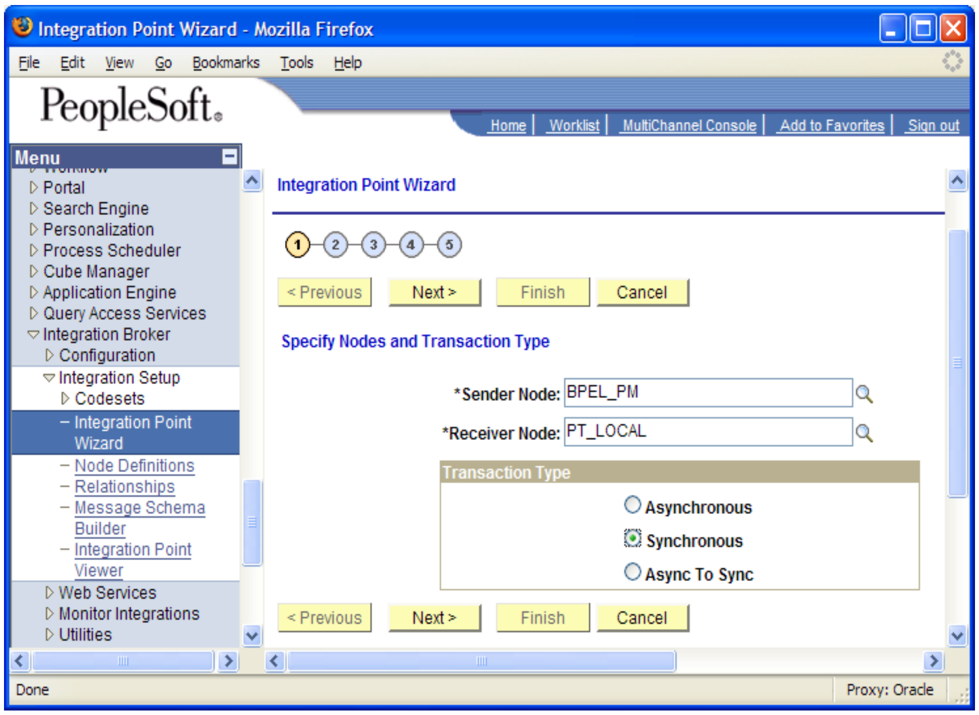
Connector ID HTTPTARGET

Properties

	*Property ID	*Property Name	Required	Value
1	HEADER	sendUncompressed	✓	Y
2	HTTPPROPERTY	Method	✓	POST
3	PRIMARYURL	URL	✓	http://<your BPEL system URL>

Click the Ping Node button on the Connectors tab to verify that Integration Broker can contact the BPEL HTTP server. If the ping does not succeed, have your administrator verify that the BPEL HTTP server is running and that you specified the correct URL and port in the node definition.

Use the Integration Point Wizard to configure the Inbound Synchronous Transaction. Advanced users can configure the Inbound Synchronous Transaction manually on the BPEL node definition.



Integration Point Wizard

1 2 3 4 5

Specify Nodes and Transaction Type

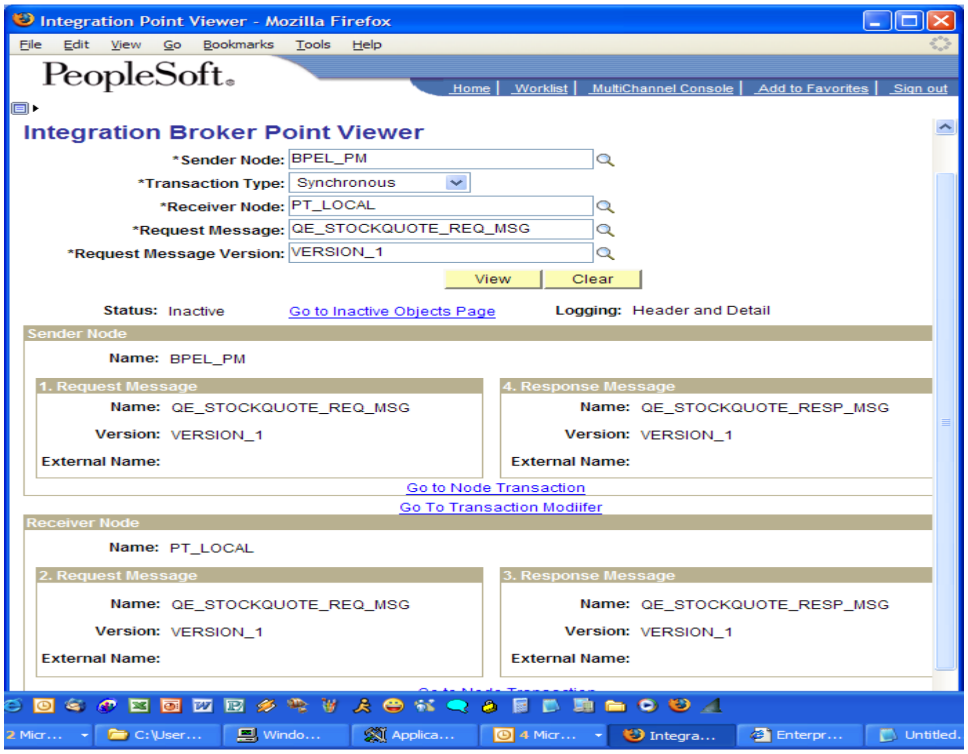
*Sender Node: BPEL_PM

*Receiver Node: PT_LOCAL

Transaction Type

Asynchronous [✓ Synchronous] Async To Sync

You can use the Integration Point Viewer to verify that you created the transaction properly.


Integration Point Viewer - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
PeopleSoft.
Home | Worklist | MultiChannel Console | Add to Favorites | Sign out
View
Clear
Go to Inactive Objects Page
Go to Node Transaction
Go To Transaction Modifier
2 Micr...
C:\User...
Windo...
Applica...
4 Micr...
Integra...
Enterpr...
Untitled

Integration Broker Point Viewer

*Sender Node: BPEL_PM

*Transaction Type: Synchronous

*Receiver Node: PT_LOCAL

*Request Message: QE_STOCKQUOTE_REQ_MSG

*Request Message Version: VERSION_1

Status: Inactive

Logging: Header and Detail

Sender Node

Name: BPEL_PM

1. Request Message

Name: QE_STOCKQUOTE_REQ_MSG

Version: VERSION_1

External Name:

4. Response Message

Name: QE_STOCKQUOTE_RESP_MSG

Version: VERSION_1

External Name:

Receiver Node

Name: PT_LOCAL

2. Request Message

Name: QE_STOCKQUOTE_REQ_MSG

Version: VERSION_1

External Name:

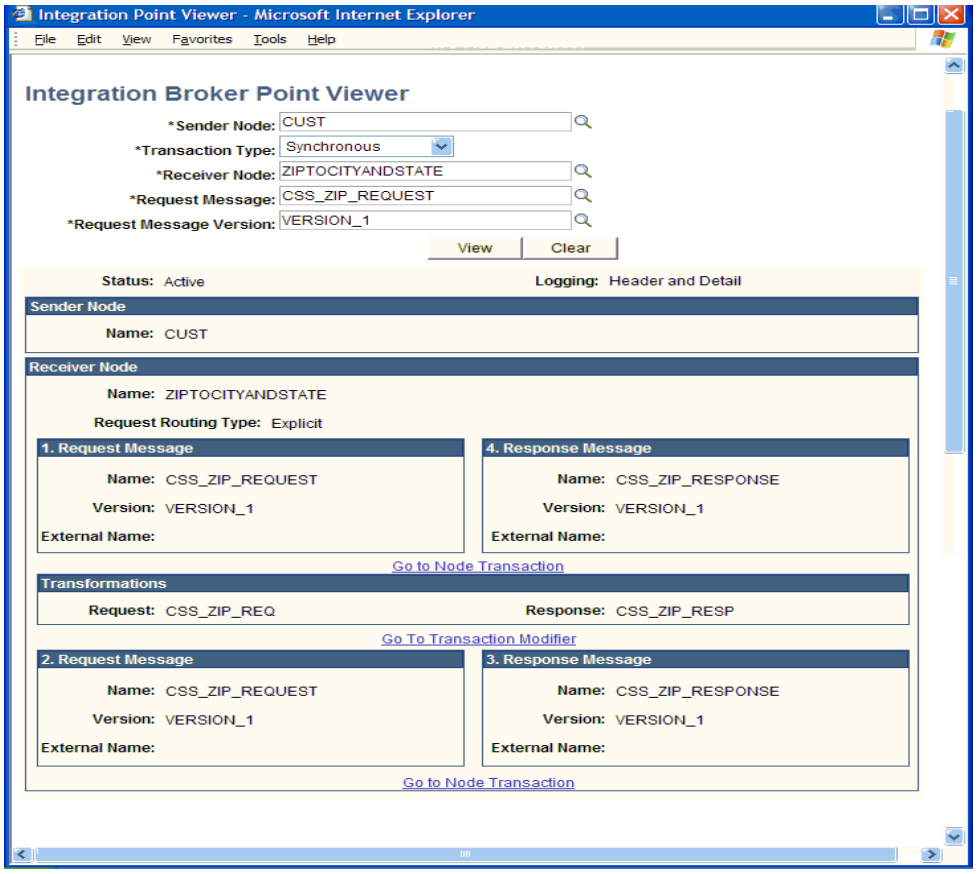
3. Response Message

Name: QE_STOCKQUOTE_RESP_MSG

Version: VERSION_1

External Name:

In the example above, BPEL_PM will send a synchronous request to PeopleSoft's PT_LOCAL node. The request message is QE_STOCKQUOTE_REQ_MSG.VERSION_1 and the response message is QE_STOCKQUOTE_RESP_MSG.VERSION_1. There are no transformations in this example. On your site, you may end up transforming an inbound SOAP request to a rowset-based message and transforming a rowset-based response to a nonrowset-based SOAP response. The Integration Point Wizard will walk you through all the steps. See the example below.


Integration Point Viewer - Microsoft Internet Explorer
File Edit View Favorites Tools Help

Integration Broker Point Viewer

*Sender Node: CUST
*Transaction Type: Synchronous
*Receiver Node: ZIPTOCITYANDSTATE
*Request Message: CSS_ZIP_REQUEST
*Request Message Version: VERSION_1

Status: Active
Logging: Header and Detail

Sender Node
Name: CUST

Receiver Node
Name: ZIPTOCITYANDSTATE
Request Routing Type: Explicit

1. Request Message
Name: CSS_ZIP_REQUEST
Version: VERSION_1
External Name:

4. Response Message
Name: CSS_ZIP_RESPONSE
Version: VERSION_1
External Name:

Go to Node Transaction

Transformations
Request: CSS_ZIP_REQ
Response: CSS_ZIP_RESP

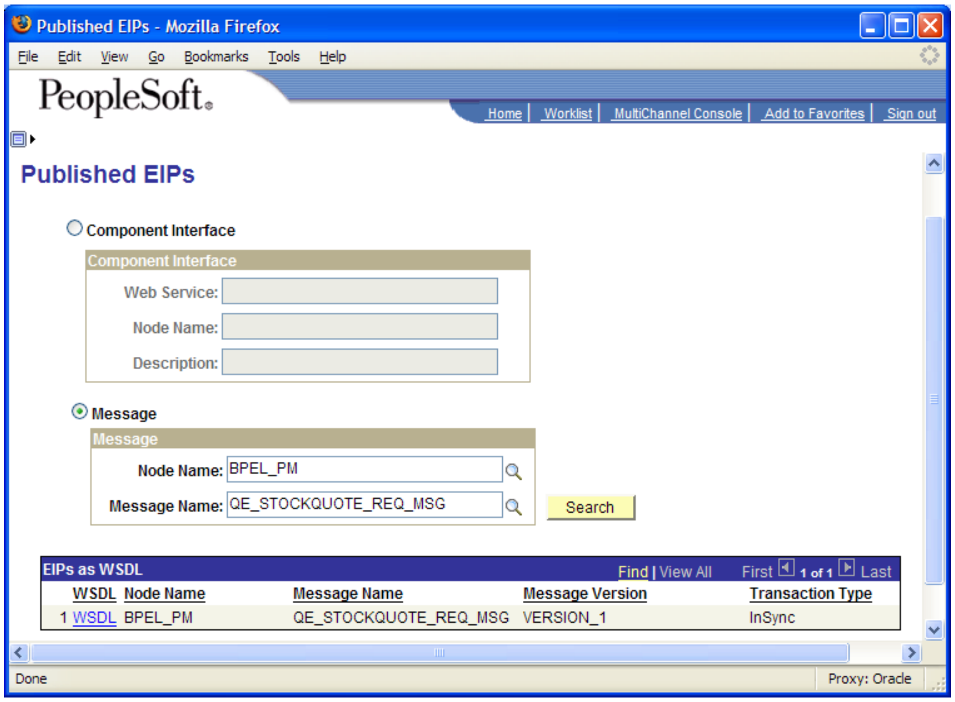
Go To Transaction Modifier

2. Request Message
Name: CSS_ZIP_REQUEST
Version: VERSION_1
External Name:

3. Response Message
Name: CSS_ZIP_RESPONSE
Version: VERSION_1
External Name:

Go to Node Transaction

With the transaction created, you are now ready to generate the WSDL for the web service. The WSDL generator can be found under Integration Broker > Web Services > Published EIPs. Select Message and then specify the node that represents your BPEL system and its request message. Then click Search.


Published EIPs - Mozilla Firefox

Home | Worklist | MultiChannel Console | Add to Favorites | Sign out

Published EIPs

( ) Component Interface

Component Interface

Web Service:

Node Name:

Description:

(●) Message

Message

Node Name: BPEL_PM

Message Name: QE_STOCKQUOTE_REQ_MSG

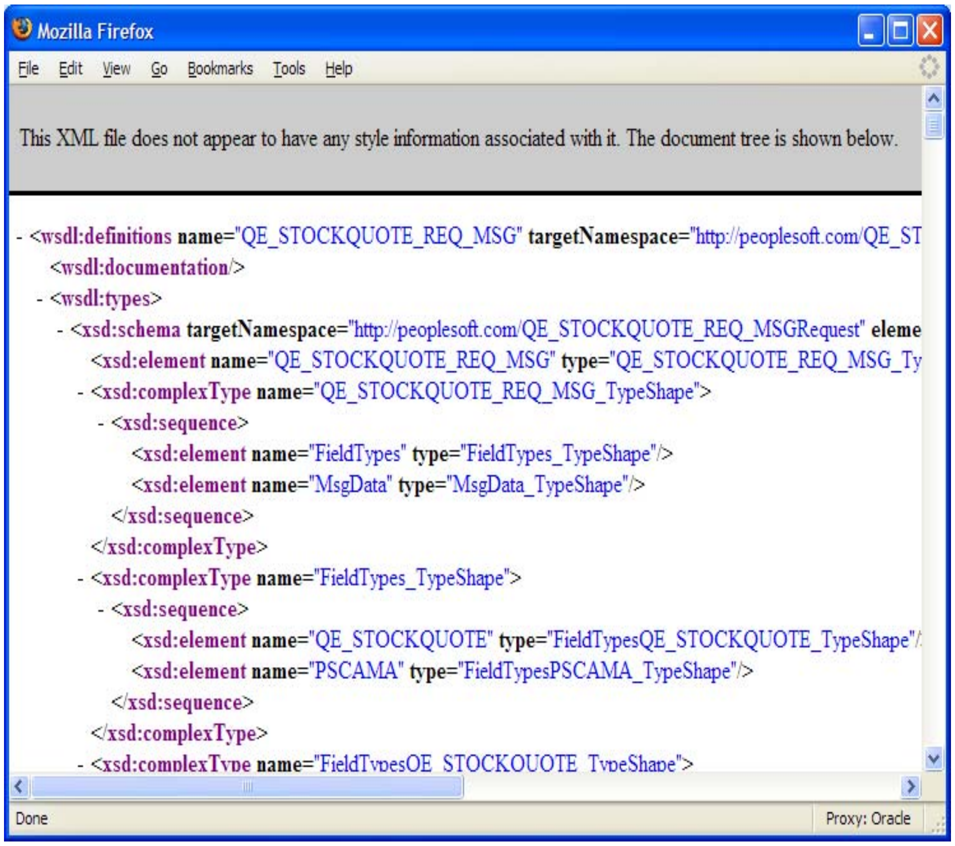
Search

EIPs as WSDL — Find | View All — First 1 of 1 Last

	WSDL	Node Name	Message Name	Message Version	Transaction Type
1	WSDL	BPEL_PM	QE_STOCKQUOTE_REQ_MSG	VERSION_1	InSync

Done — Proxy: Oracle

Click the WSDL link to generate the WSDL. A new browser will open with the WSDL. You can use either just the URL in the new browser window for BPEL PM to consume the WSDL, or you can save the WSDL as a file for BPEL PM to browse.



This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
- <wsdl:definitions name="QE_STOCKQUOTE_REQ_MSG" targetNamespace="http://peoplesoft.com/QE_ST
    <wsdl:documentation/>
  - <wsdl:types>
    - <xsd:schema targetNamespace="http://peoplesoft.com/QE_STOCKQUOTE_REQ_MSGRequest" eleme
        <xsd:element name="QE_STOCKQUOTE_REQ_MSG" type="QE_STOCKQUOTE_REQ_MSG_Ty
      - <xsd:complexType name="QE_STOCKQUOTE_REQ_MSG_TypeShape">
        - <xsd:sequence>
            <xsd:element name="FieldTypes" type="FieldTypes_TypeShape"/>
            <xsd:element name="MsgData" type="MsgData_TypeShape"/>
          </xsd:sequence>
        </xsd:complexType>
      - <xsd:complexType name="FieldTypes_TypeShape">
        - <xsd:sequence>
            <xsd:element name="QE_STOCKQUOTE" type="FieldTypesQE_STOCKQUOTE_TypeShape"/
            <xsd:element name="PSCAMA" type="FieldTypesPSCAMA_TypeShape"/>
          </xsd:sequence>
        </xsd:complexType>
      - <xsd:complexType name="FieldTypesQE_STOCKQUOTE_TypeShape">
```

The WSDL is now ready to be consumed by BPEL PM.

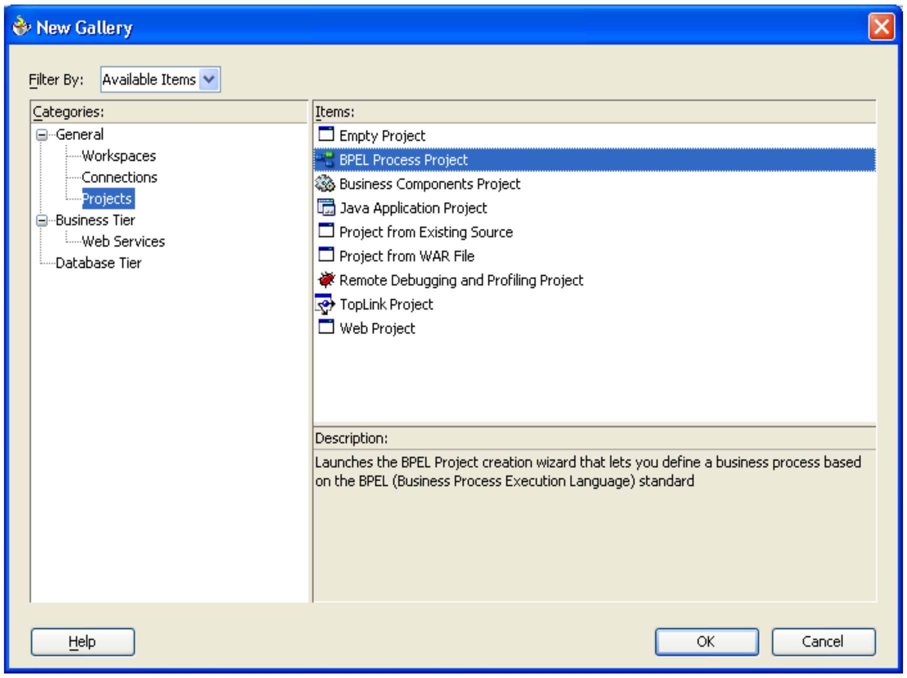
Generating BPEL PM process to consume the service

The following steps assume that BPEL PM server and designer are properly set up and that the appropriate workspace and connections are created in the designer.

Start BPEL PM server.

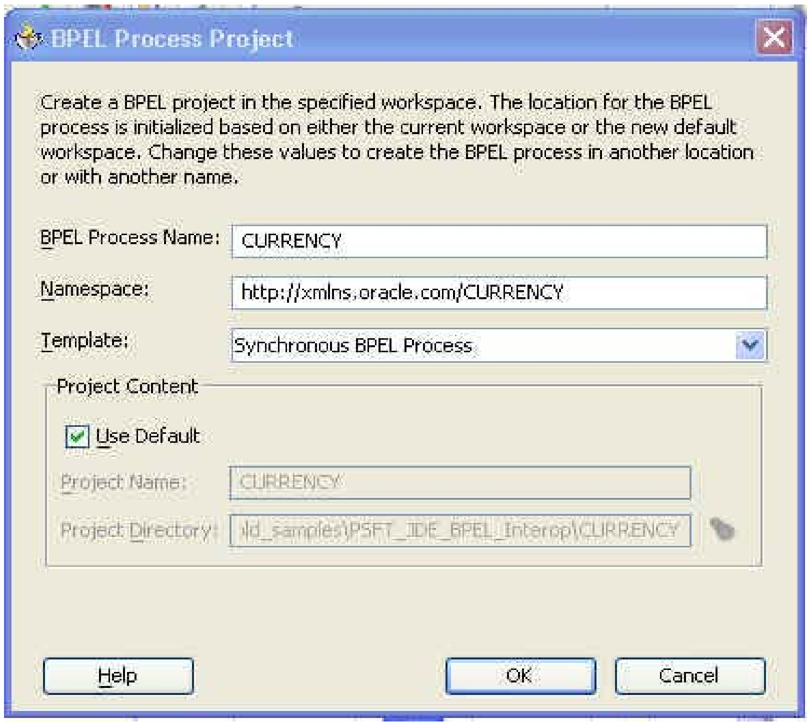
Launch BPEL Designer. Right-click on the workspace and choose New Project.

From Project Items, choose BPEL Process Project and click OK.



Description:
Launches the BPEL Project creation wizard that lets you define a business process based on the BPEL (Business Process Execution Language) standard

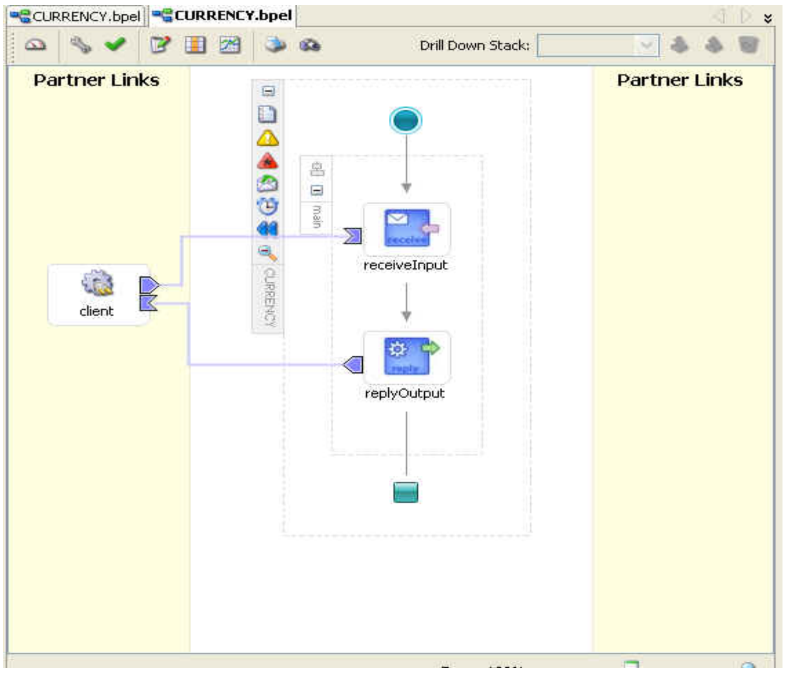
In the project properties dialog, select process name, and choose Synchronous service from Template dropdown. Click OK.



Create a BPEL project in the specified workspace. The location for the BPEL process is initialized based on either the current workspace or the new default workspace. Change these values to create the BPEL process in another location or with another name.

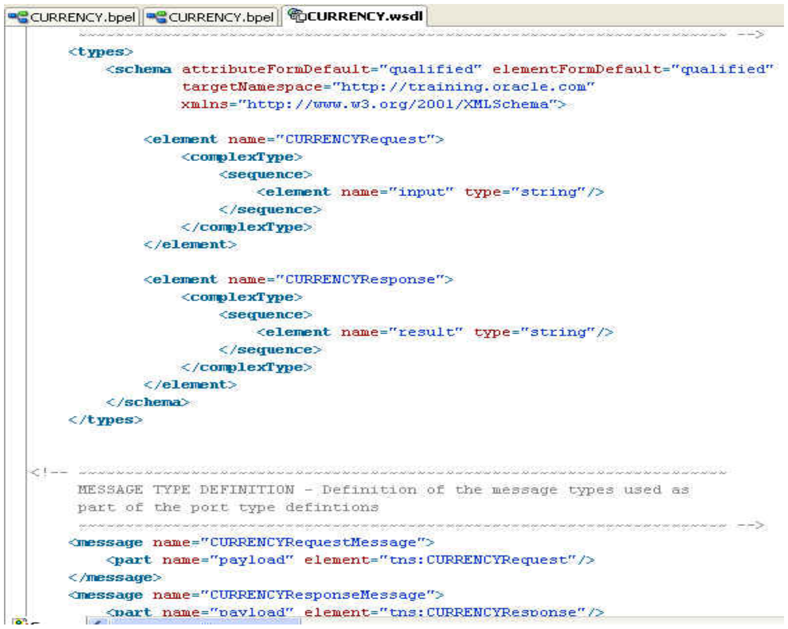
BPEL Process Name: CURRENCY
Namespace: http://xmlns.oracle.com/CURRENCY
Template: Synchronous BPEL Process
[✓] Use Default
Project Name: CURRENCY
Project Directory: ld_samples\PSFT_IDE_BPEL_Interop\CURRENCY

The following sections appear. If this does not appear, click Diagram View and double click CURRENCY.bpel in the Applications Navigator section. See the *Oracle BPEL Process Manager Developer's Guide* for a description of these sections.



Browse to the project directory from explorer and copy the WSDL generated from WSG Developer to this folder.

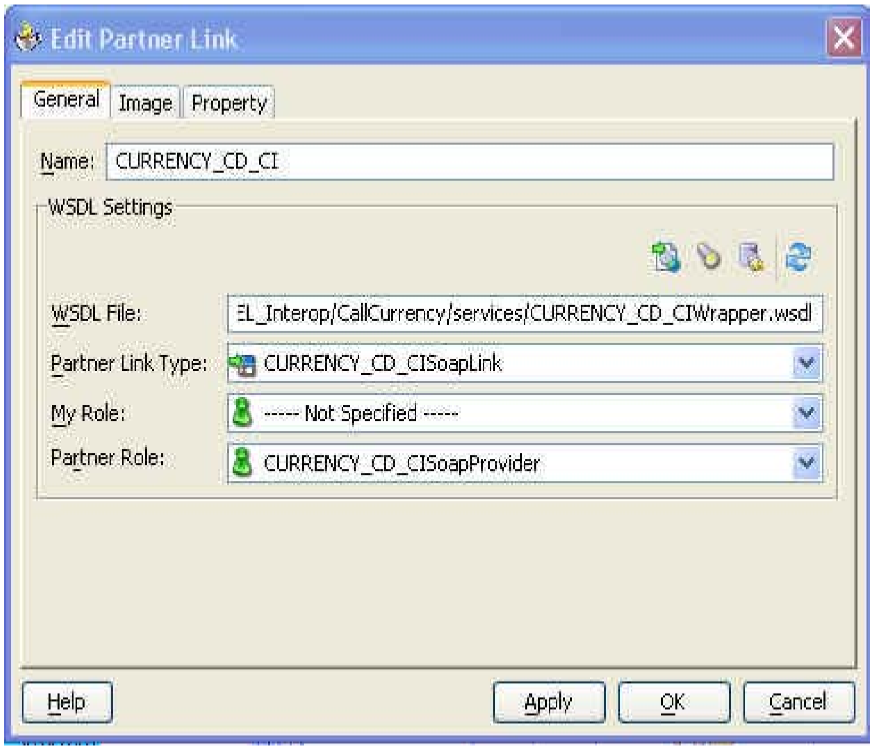
In Applications –Navigator, navigate to <processname>.WSDL file under Integration Content. Double-click to open the file and view WSDL source.



```
     ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ -->
    <types>
        <schema attributeFormDefault="qualified" elementFormDefault="qualified"
                targetNamespace="http://training.oracle.com"
                xmlns="http://www.w3.org/2001/XMLSchema">

            <element name="CURRENCYRequest">
                <complexType>
                    <sequence>
                        <element name="input" type="string"/>
                    </sequence>
                </complexType>
            </element>

            <element name="CURRENCYResponse">
                <complexType>
                    <sequence>
                        <element name="result" type="string"/>
                    </sequence>
                </complexType>
            </element>
        </schema>
    </types>


<!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     MESSAGE TYPE DEFINITION - Definition of the message types used as
     part of the port type defintions
     ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ -->
    <message name="CURRENCYRequestMessage">
        <part name="payload" element="tns:CURRENCYRequest"/>
    </message>
    <message name="CURRENCYResponseMessage">
        <part name="payload" element="tns:CURRENCYResponse"/>
```

Now double-click on ".bpel" file to open it in the editor.

Right-click on either side of the BPEL process (yellow area) and choose **create partner link** option.



Name: CURRENCY_CD_CI

WSDL Settings

WSDL File: EL_Interop/CallCurrency/services/CURRENCY_CD_CIWrapper.wsdl

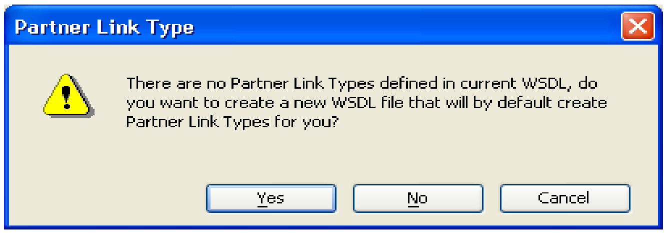
Partner Link Type: CURRENCY_CD_CISoapLink

My Role: ----- Not Specified -----

Partner Role: CURRENCY_CD_CISoapProvider

Provide a partner link name. Click on the first icon in the WSDL Settings frame to select the WSDL file from file system. Select the WSG WSDL you copied to the project directory.

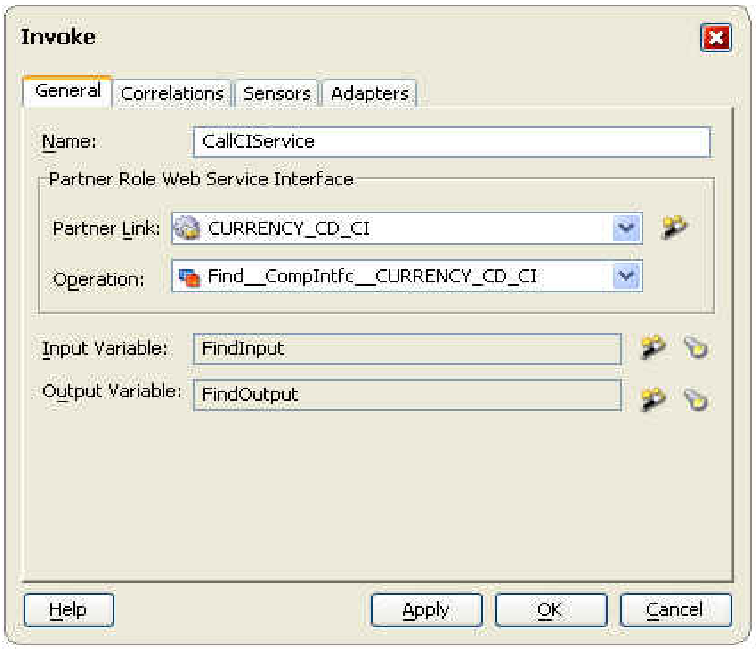
The Designer will display the following message:



There are no Partner Link Types defined in current WSDL, do you want to create a new WSDL file that will by default create Partner Link Types for you?

Click Yes to automatically create partner link types. Designer will create the partner link type and populate it. Keep My Roles filed as Not specified and populate Partner Role with the value available in the drop down. Click OK to close the partner link setting and save the bpel file.

Drag and drop an "invoke" BPEL activity to the Main scope and point it to the Partner Link configured in the previous step. The Input and Output Variables to this "invoke" activity are Global variables and are named as "FindInput" and "FindOutput" in the example. Please refer to the *BPEL Developer's Guide* for detailed steps. The CI WSDL contains many different operations – "Find", "Get", "Create" and "Update". This example uses the "Find" operation.



Name: CallCIService

Partner Role Web Service Interface

Partner Link: CURRENCY_CD_CI

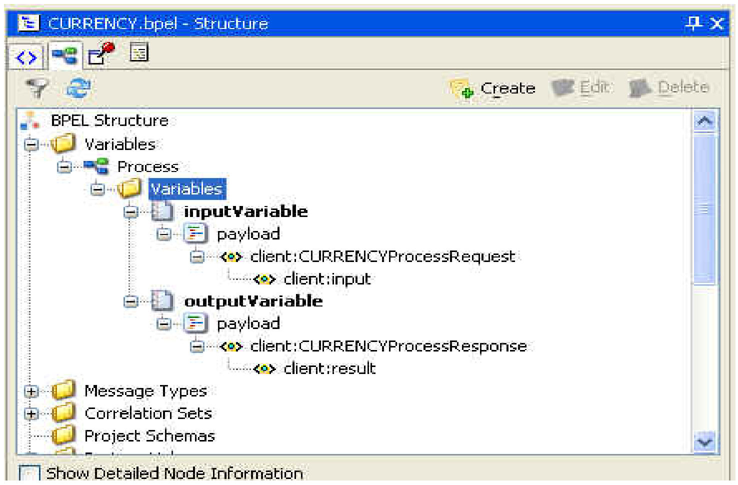
Operation: Find__CompIntfc__CURRENCY_CD_CI

Input Variable: FindInput

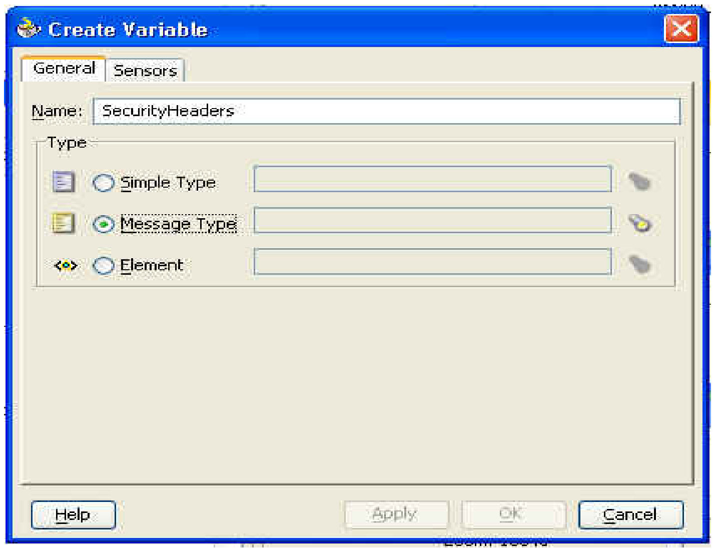
Output Variable: FindOutput

```
<wsdl:operation name="Find__CompIntfc__CURRENCY_CD_CI">
    <soap:operation style="document"
soapAction="http://peoplesoft.com/SOAPTOCI/SOAP_NODE//QE_LOCAL"/>
    <wsdl:input>
          <soap:body use="literal" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
<soap:header use="literal" part="parameter"
            message="wsdl_target:SecurityHeaders"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
        </wsdl:input>
    <wsdl:output>
          <soap:body use="literal" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
    </wsdl:output>
    <wsdl:fault name="Fault">
          <soap:fault use="literal" name="Fault"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
    </wsdl:fault>
  </wsdl:operation>
```

The above operation takes on a <*SecurityHeader*> element in addition to the input. Create a Global Variable to point to the <*SecurityHeaders*> element in the Partner WSDL as shown in the following steps:

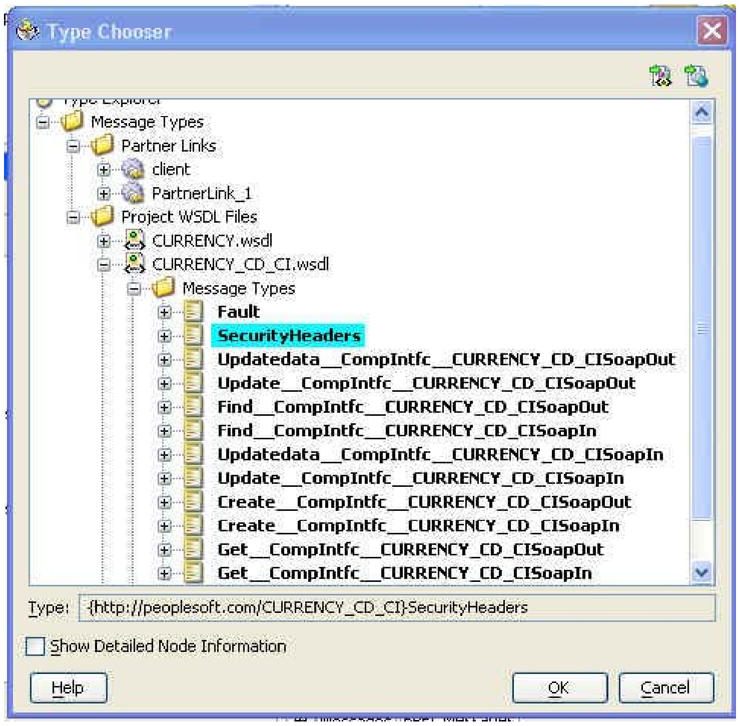


Go to the Variables folder on the BPEL Structure Pane and right-click it. Select the "Create" option and point it to <*SecurityHeaders*> msg in the CI WSDL.



Name: SecurityHeaders

Simple Type [✓ Message Type] Element


Type Chooser

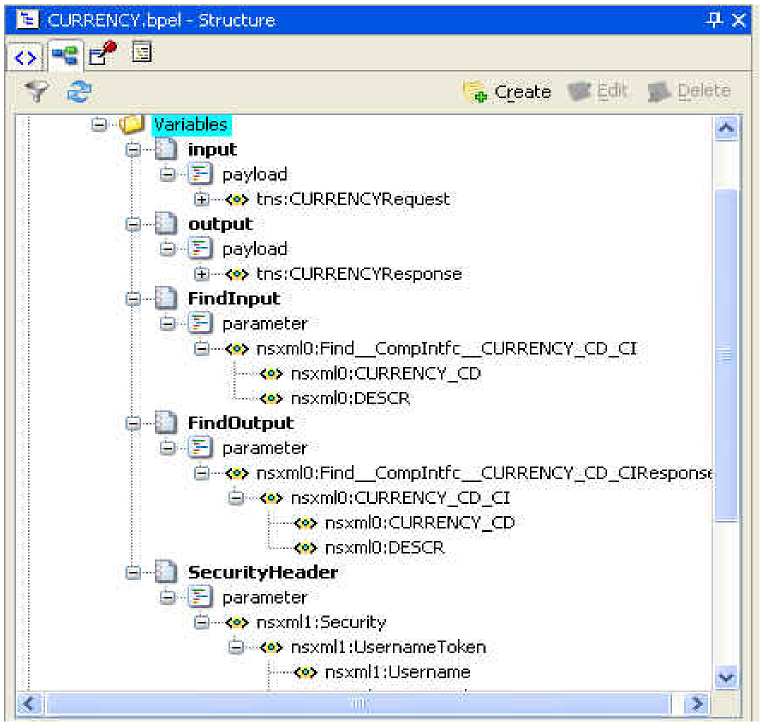
Type Explorer
- Message Types
  - Partner Links
    - client
    - PartnerLink_1
  - Project WSDL Files
    - CURRENCY.wsdl
    - CURRENCY_CD_CI.wsdl
      - Message Types
        - Fault
        - SecurityHeaders
        - Updatedata__CompIntfc__CURRENCY_CD_CISoapOut
        - Update__CompIntfc__CURRENCY_CD_CISoapOut
        - Find__CompIntfc__CURRENCY_CD_CISoapOut
        - Find__CompIntfc__CURRENCY_CD_CISoapIn
        - Updatedata__CompIntfc__CURRENCY_CD_CISoapIn
        - Update__CompIntfc__CURRENCY_CD_CISoapIn
        - Create__CompIntfc__CURRENCY_CD_CISoapOut
        - Create__CompIntfc__CURRENCY_CD_CISoapIn
        - Get__CompIntfc__CURRENCY_CD_CISoapOut
        - Get__CompIntfc__CURRENCY_CD_CISoapIn

Type: {http://peoplesoft.com/CURRENCY_CD_CI}SecurityHeaders

Show Detailed Node Information

Help OK Cancel

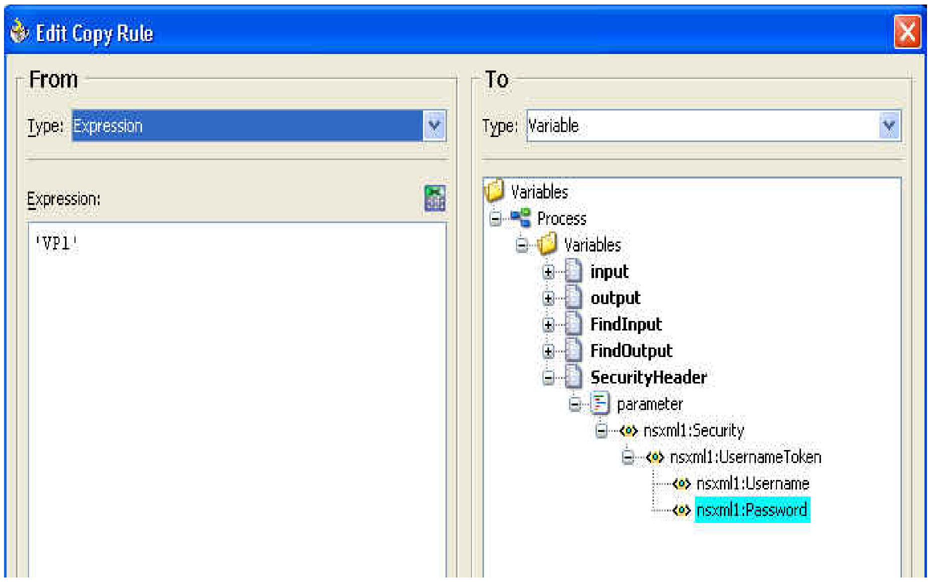
The structure of the various variables created in the BPEL process are shown below:


CURRENCY.bpel - Structure

Create Edit Delete

- Variables
  - input
    - payload
      - tns:CURRENCYRequest
  - output
    - payload
      - tns:CURRENCYResponse
  - FindInput
    - parameter
      - nsxml0:Find__CompIntfc__CURRENCY_CD_CI
        - nsxml0:CURRENCY_CD
        - nsxml0:DESCR
  - FindOutput
    - parameter
      - nsxml0:Find__CompIntfc__CURRENCY_CD_CIResponse
        - nsxml0:CURRENCY_CD_CI
          - nsxml0:CURRENCY_CD
          - nsxml0:DESCR
  - SecurityHeader
    - parameter
      - nsxml1:Security
        - nsxml1:UsernameToken
          - nsxml1:Username

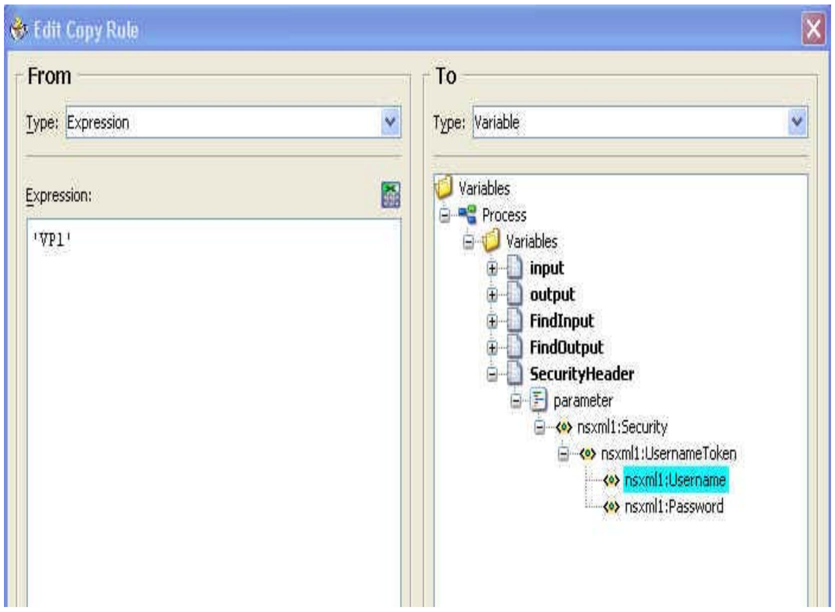
Drag and drop "assign" activities from the RHS menu to the main panel. Configure the <*SecurityHeaders*> variable as shown below.



From Type: Expression
Expression:

```
'VP1'
```

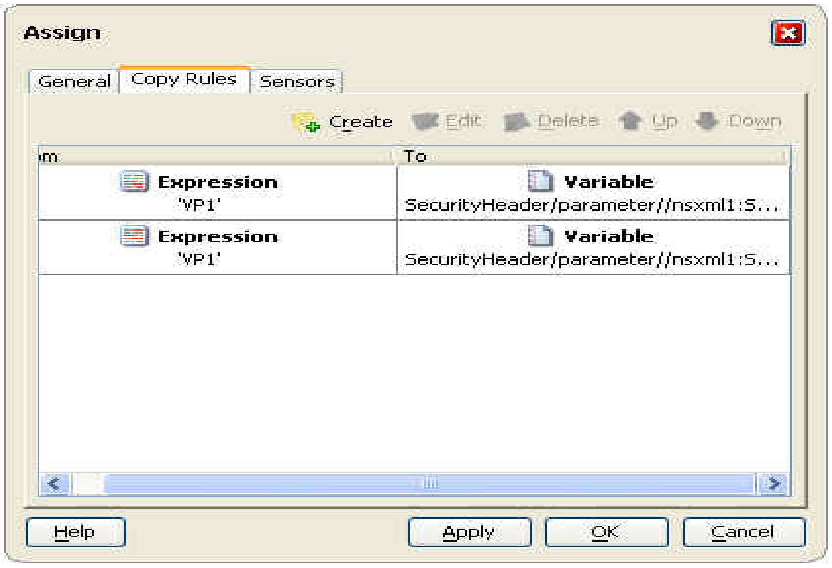
To Type: Variable



From Type: Expression
Expression:

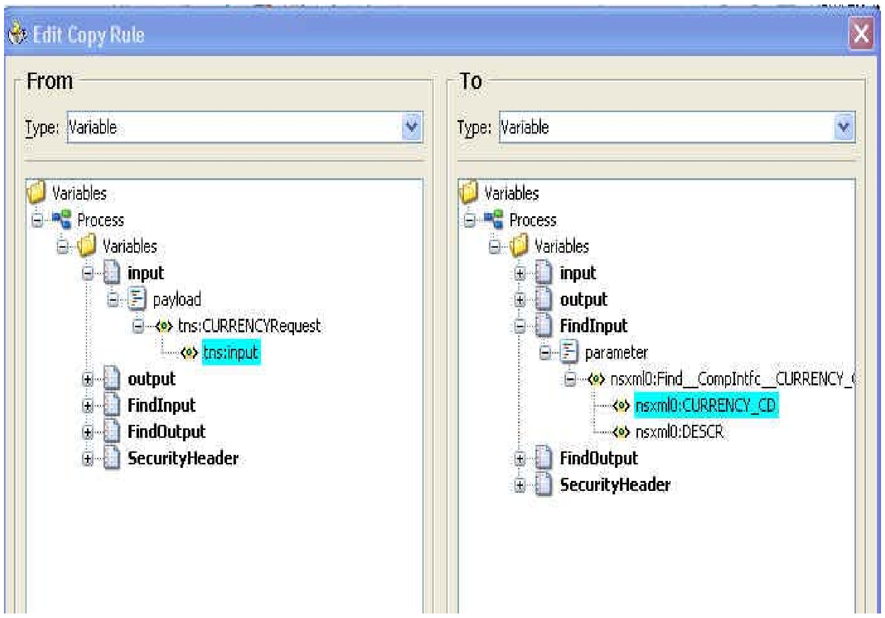
```
'VP1'
```

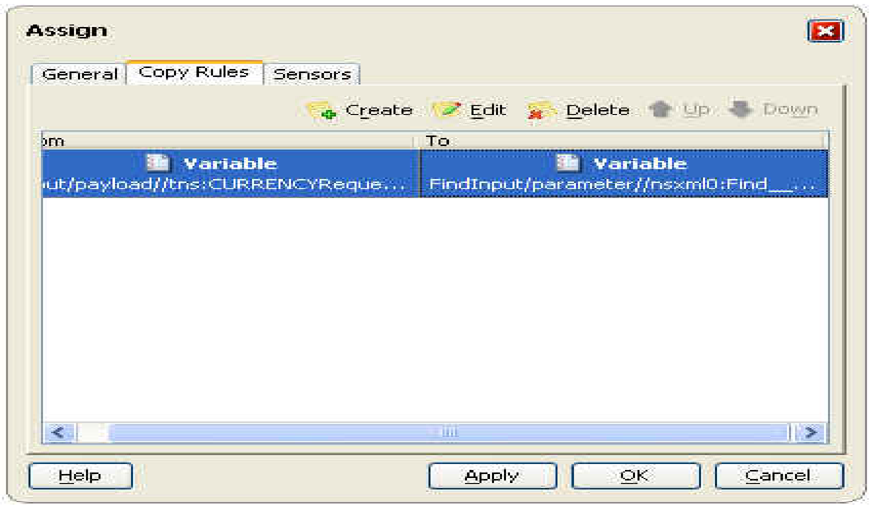
To Type: Variable



m	To
Expression 'VP1'	Variable SecurityHeader/parameter//nsxml1:S...
Expression 'VP1'	Variable SecurityHeader/parameter//nsxml1:S...

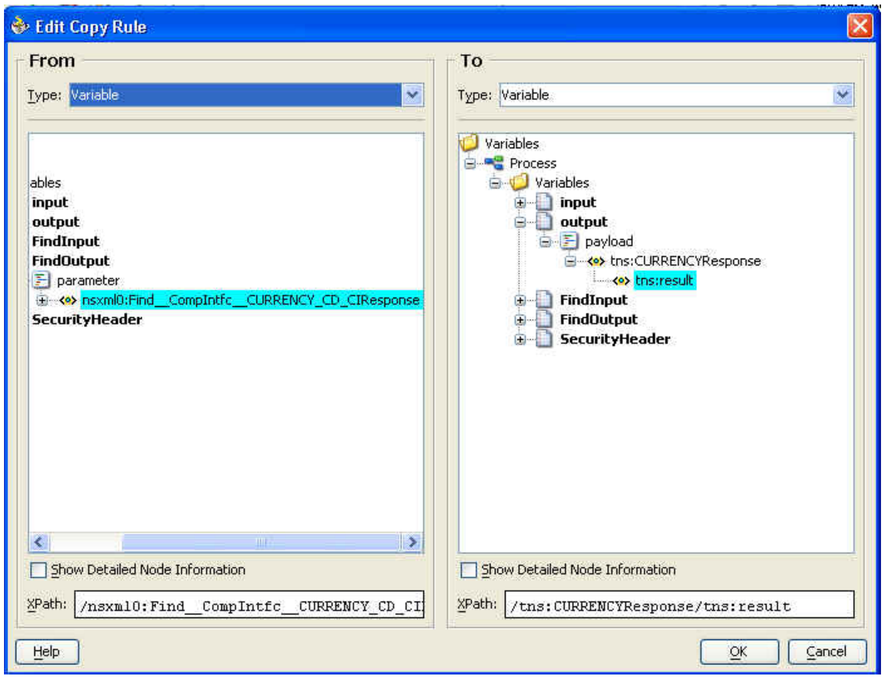
Configure the <FindInput> variable by dragging and dropping an "assign" activity. Steps are captured below:





om	To
Variable ut/payload//tns:CURRENCYReque...	Variable FindInput/parameter//nsxml0:Find__...

Configure the <FindOutput> variable by dragging and dropping a third assign activity as shown below:

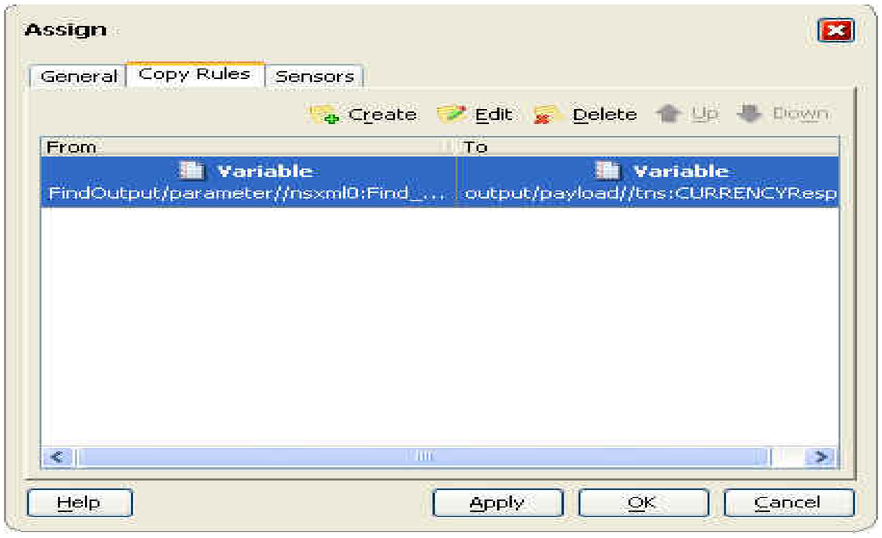


From Type: Variable

XPath: /nsxml0:Find__CompIntfc__CURRENCY_CD_CI

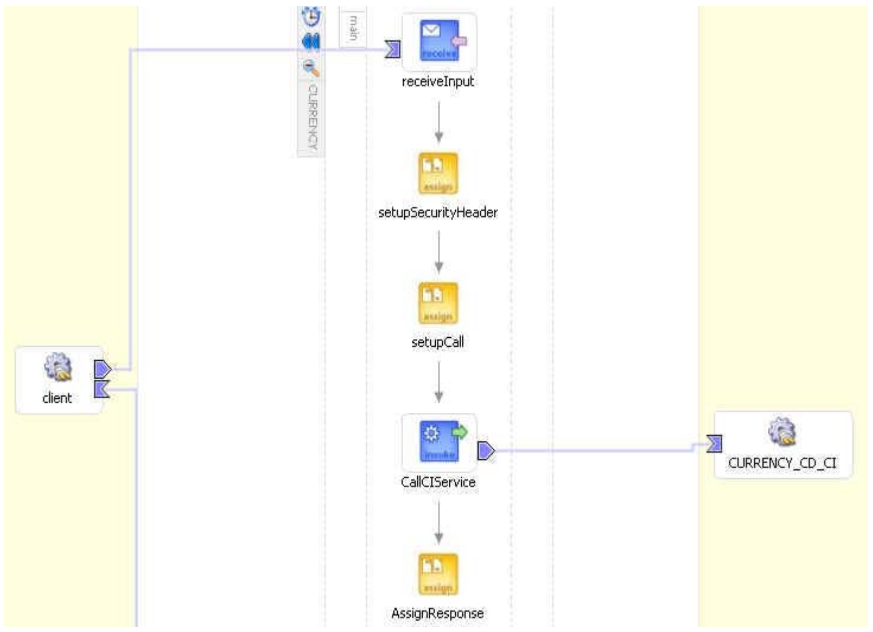
To Type: Variable

XPath: /tns:CURRENCYResponse/tns:result



From	To
Variable FindOutput/parameter//nsxml0:Find_...	Variable output/payload//tns:CURRENCYResp

The entire BPEL process is captured below:



To compile and Deploy the BPEL process, right-click on the project in Application Navigator. Select Deploy > *connection name* > Deploy to Default Domain. Click OK. This will compile and deploy the BPEL process. Watch the bottom of the screen for any errors. Refer to the *BPEL Developer's Guide* for more information.

BPELconsuming AM

PeopleSoft Providing Message Based Aynchronous Web Services

Asynchronous messages can be exposed via web services, regardless if they are rowset-based (structured) or nonrowset-based (unstructured). This document explains how to web service enable Application Messages as asynchronous web services. With asynchronous web services, either the consumer does not require a response, or an acknowledgment message will be sent back at a later time.

Note: If an unstructured message does not have an associated schema, the schema section of the WSDL document will be incomplete. You must either associate a schema with the message or modify the schema section of the WSDL so that the WSDL document completely describes the message. If you are using a delivered unstructured message where a schema has been associated, that schema will appear in the WSDL.

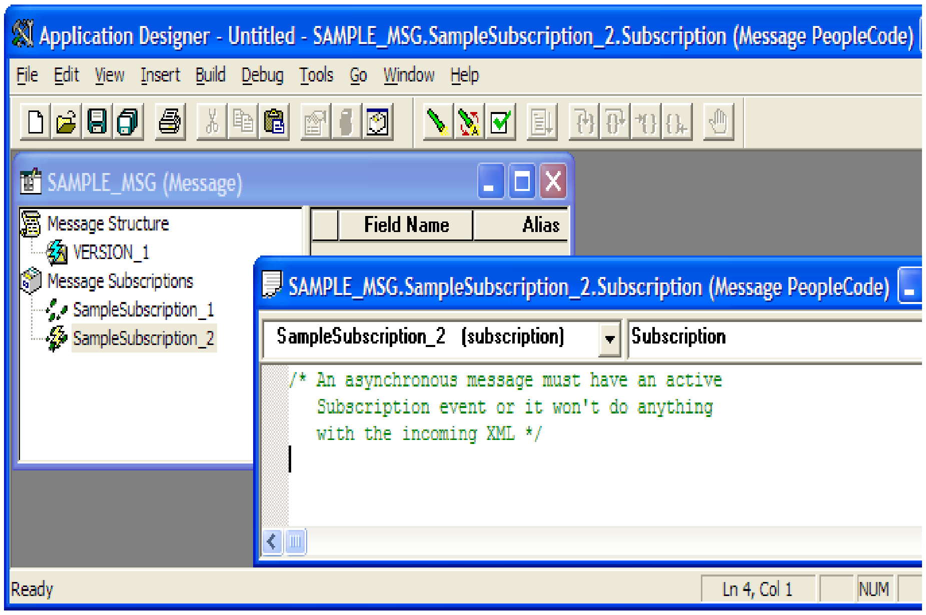
This functionality uses the basic PeopleSoft Integration Broker infrastructure with one exception: at runtime SOAP request messages are sent to a specific listening connector on the gateway. External parties send web service requests to the PeopleSoftServiceListeningConnector, which is responsible for stripping the SOAP envelope and forwarding the request to the application server for processing. The application server receives the request as it would any other message, and invokes the appropriate message processing.

The primary differences between CI and message based web services are:

- In a CI based web service, all of the logic already exists in the component. You do not have to add any business logic to a CI.
- In Application message based web services, if there is no delivered logic in a message event, you must code the logic yourself.
- To invoke a CI based web service, the invoking system must always supply a valid PeopleSoft username and password, with the correct permissions, in order to invoke the specified service.

Application Message based web services never take a username and password. They can be secured, however, using digital certificates, SSL, or a node password. For more information see Enterprise PeopleTools 8.46 PeopleBook: Integration Broker > Setting Up Secure Messaging Environments.

For asynchronous message based web services, the request message must have a Subscription event. See below:

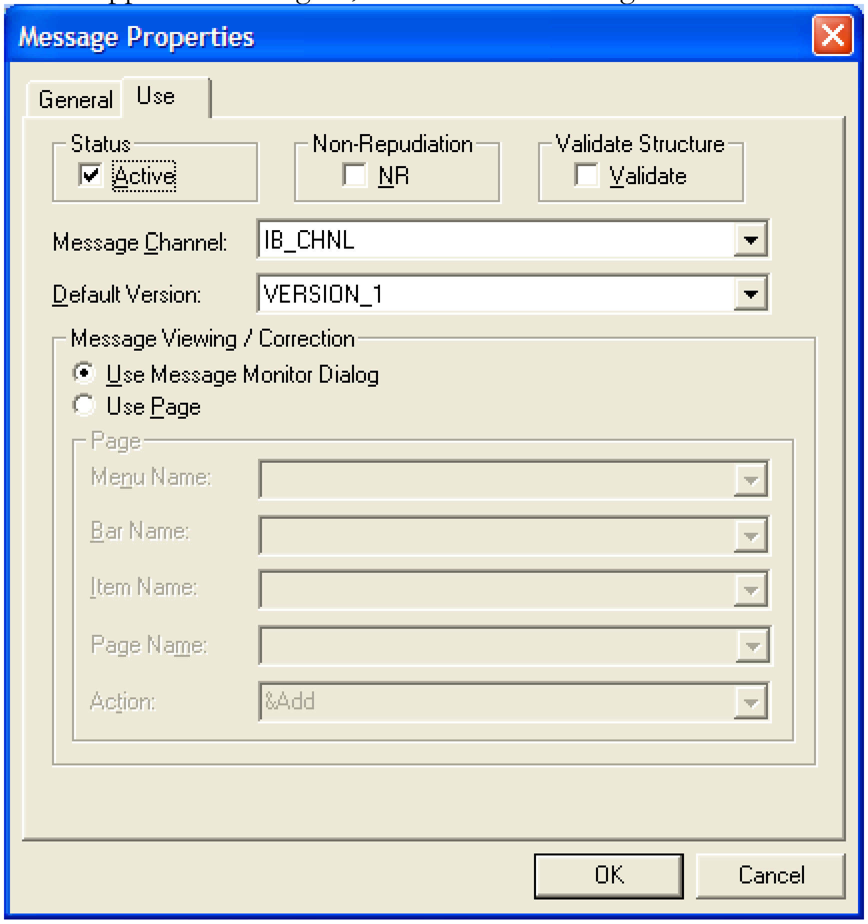

Application Designer - Untitled - SAMPLE_MSG.SampleSubscription_2.Subscription (Message PeopleCode)
File Edit View Insert Build Debug Tools Go Window Help
SAMPLE_MSG (Message)
Message Structure
VERSION_1
Message Subscriptions
SampleSubscription_1
SampleSubscription_2
Field Name
Alias
SAMPLE_MSG.SampleSubscription_2.Subscription (Message PeopleCode)
SampleSubscription_2 (subscription)
Subscription
Ready
Ln 4, Col 1
NUM

```
/* An asynchronous message must have an active
   Subscription event or it won't do anything
   with the incoming XML */
```

The Subscription event contains or calls the code to read and do something with the incoming XML payload. The code may or may not send back a response message. You can have more than one subscription active but there is no way to designate or determine which one will fire first.

To Web Service Enable Asynchronous Messages

In the Application Designer, make sure the message is Active.


Message Properties

General | Use

Status: [✓ Active]

Non-Repudiation: [ ] NR

Validate Structure: [ ] Validate

Message Channel: IB_CHNL

Default Version: VERSION_1

Message Viewing / Correction

(●) Use Message Monitor Dialog

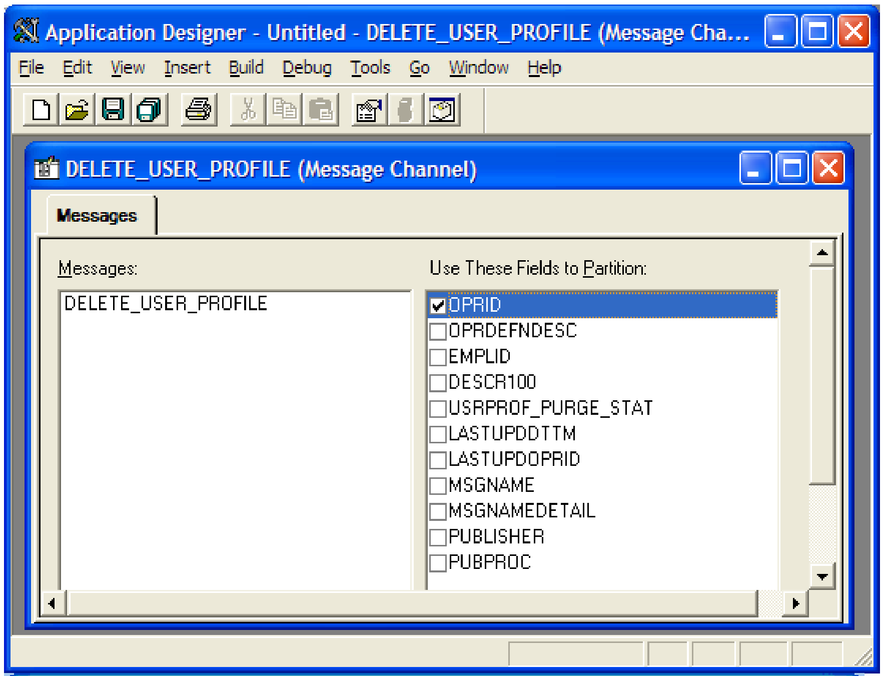
( ) Use Page

Page

Menu Name:	
Bar Name:	
Item Name:	
Page Name:	
Action:	&Add

If message processing order is important, make sure the message is assigned to a channel that you can partition by the appropriate process-order-determining field. If a channel is set to use ordered processing (default) and you have not selected any partitioning fields (default), sever bottlenecking performance issues can result.

In the example below, the DELETE_USER_PROFILE channel is partitioned on the OPRID (primary key) field. If a message for a particular OPRID errors, messages for all other OPRIDs will continue to be processed. If no field was selected for partitioning, then if any message would error then all processing would be stopped for all new messages to enter the channel until the original error is resolved.


Application Designer - Untitled - DELETE_USER_PROFILE (Message Cha...

DELETE_USER_PROFILE (Message Channel)

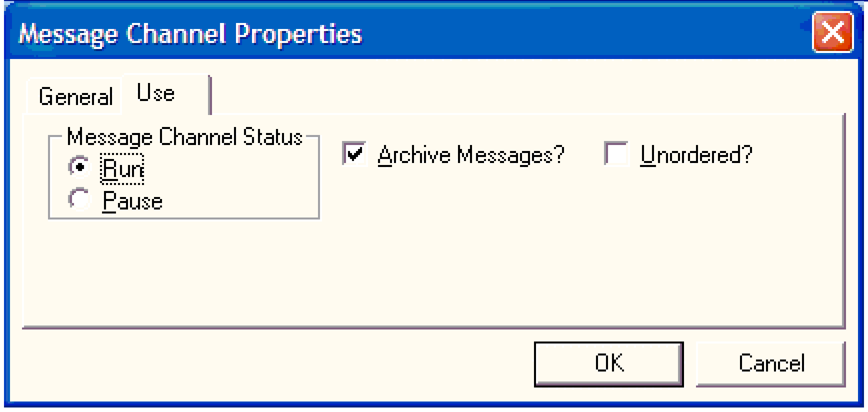
Messages

Messages:

DELETE_USER_PROFILE

Use These Fields to Partition:

- [✓] OPRID
- [ ] OPRDEFNDESC
- [ ] EMPLID
- [ ] DESCR100
- [ ] USRPROF_PURGE_STAT
- [ ] LASTUPDDTTM
- [ ] LASTUPDOPRID
- [ ] MSGNAME
- [ ] MSGNAMEDETAIL
- [ ] PUBLISHER
- [ ] PUBPROC


Message Channel Properties

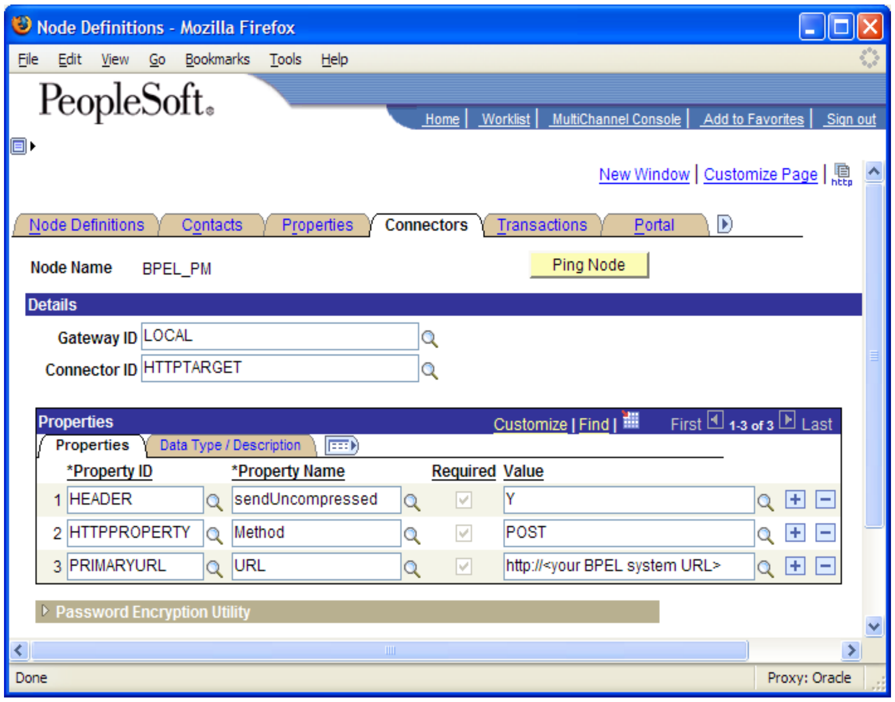
General | Use

Message Channel Status: [✓ Run] Pause

[✓] Archive Messages?

[ ] Unordered?

In the Integration Broker Node Definitions, either verify the existence of or create a node definition for the BPEL PM system.


Node Definitions - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
PeopleSoft.
Home | Worklist | MultiChannel Console | Add to Favorites | Sign out
New Window | Customize Page
Node Definitions
Contacts
Properties
Connectors
Transactions
Portal
Ping Node
Details
Customize | Find | First 1-3 of 3 Last
Properties
Data Type / Description
Password Encryption Utility
Done
Proxy: Oracle

Node Name BPEL_PM

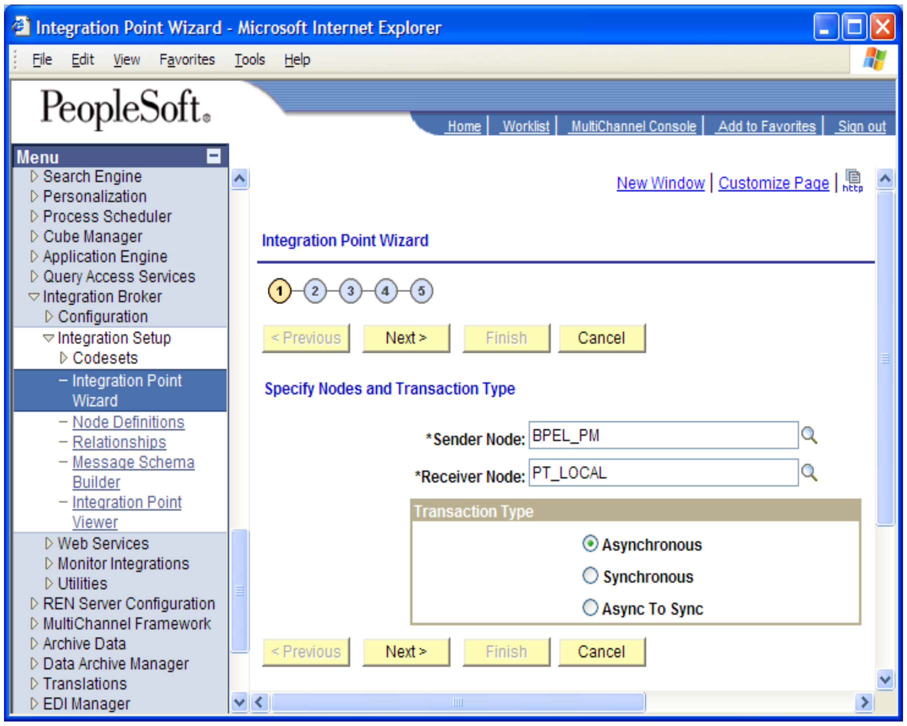
Gateway ID LOCAL

Connector ID HTTPTARGET

	*Property ID	*Property Name	Required	Value
1	HEADER	sendUncompressed	✓	Y
2	HTTPPROPERTY	Method	✓	POST
3	PRIMARYURL	URL	✓	http://<your BPEL system URL>

Click the Ping Node button on the Connectors tab to verify that Integration Broker can contact the BPEL HTTP server. If the ping does not succeed, have your administrator verify that the BPEL HTTP server is running and that you specified the correct URL and port in the node definition.

Use the Integration Point Wizard to configure the Inbound Asynchronous Transaction. Advanced users can configure the Inbound Asynchronous Transaction manually on the BPEL node definition.



Integration Point Wizard

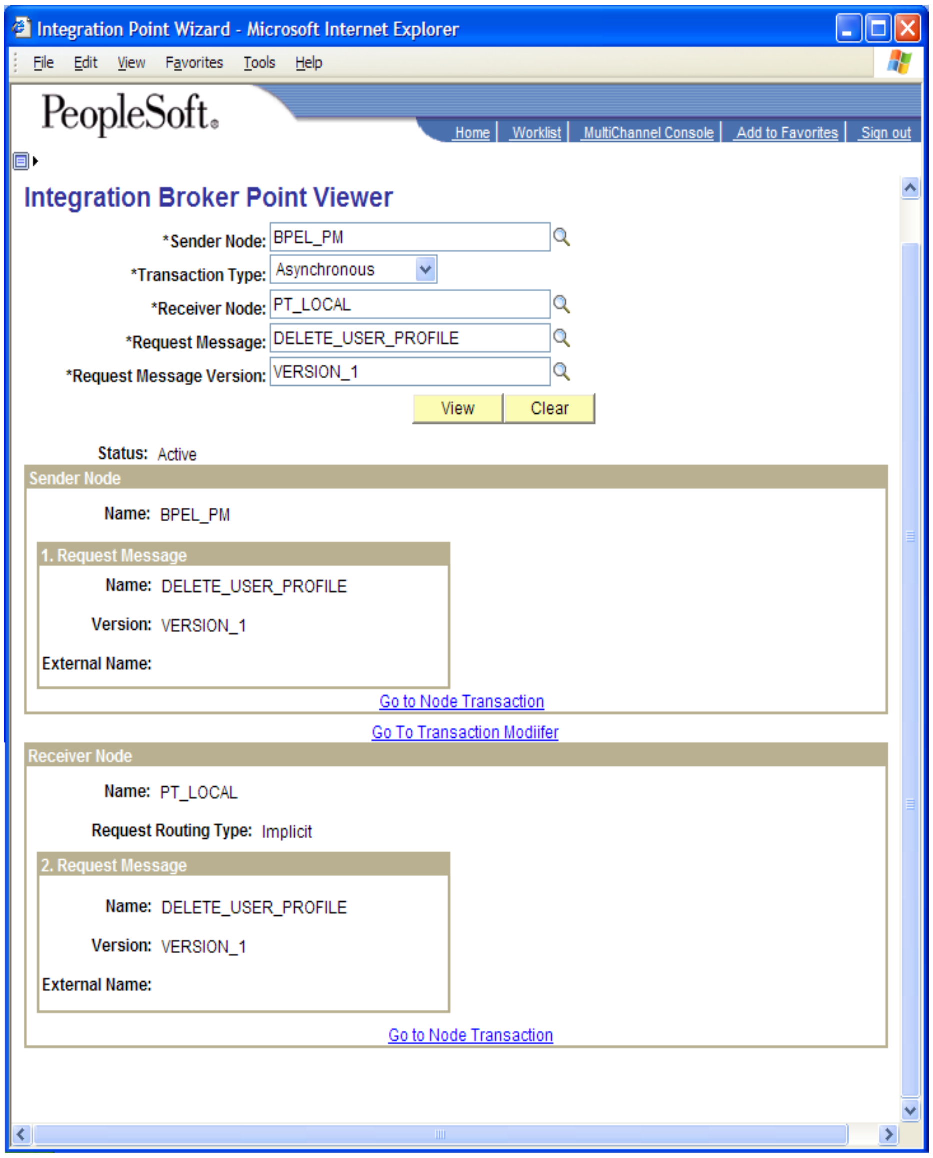
Specify Nodes and Transaction Type

*Sender Node: BPEL_PM

*Receiver Node: PT_LOCAL

Transaction Type: [✓ Asynchronous] Synchronous Async To Sync

You can use the Integration Point Viewer to verify that you created the transaction properly.


Integration Point Wizard - Microsoft Internet Explorer
File Edit View Favorites Tools Help
PeopleSoft.
Home | Worklist | MultiChannel Console | Add to Favorites | Sign out
View
Clear
Go to Node Transaction
Go To Transaction Modiifer
Go to Node Transaction

Integration Broker Point Viewer

*Sender Node: BPEL_PM
*Transaction Type: Asynchronous
*Receiver Node: PT_LOCAL
*Request Message: DELETE_USER_PROFILE
*Request Message Version: VERSION_1

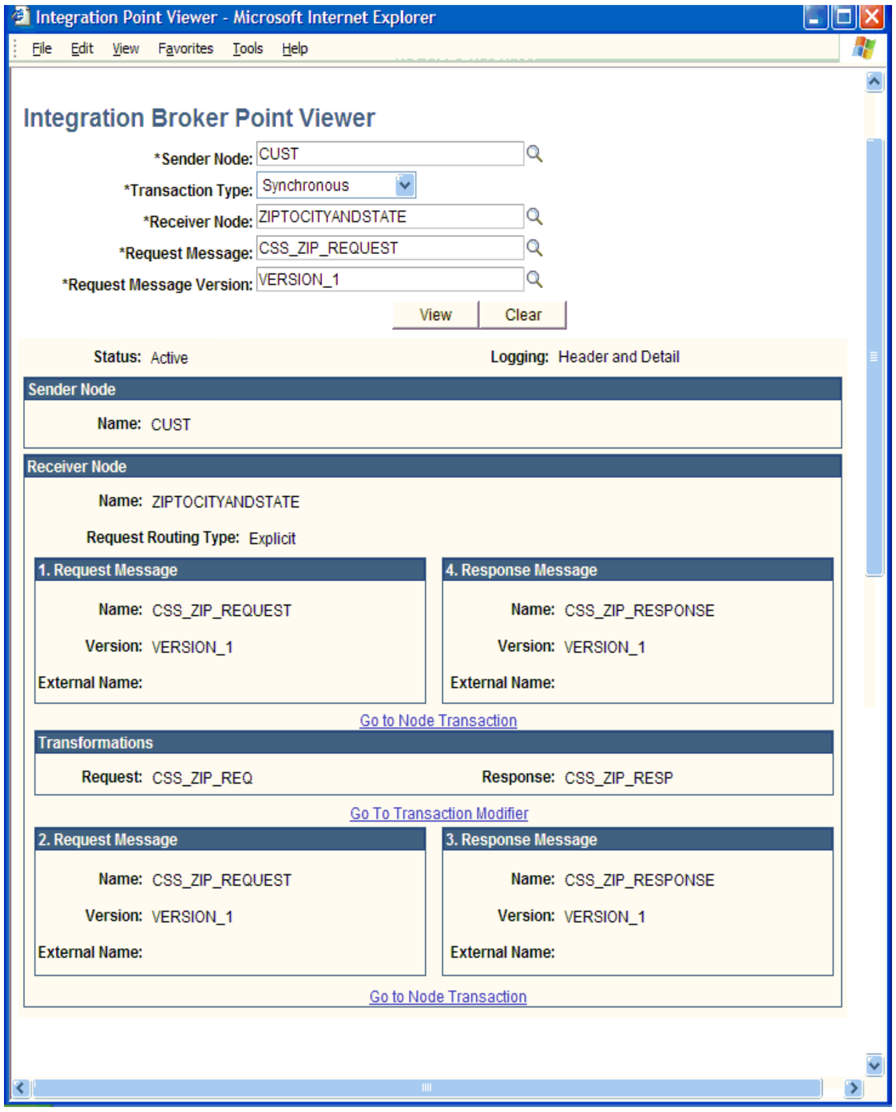
Status: Active

Sender Node
Name: BPEL_PM
1. Request Message
Name: DELETE_USER_PROFILE
Version: VERSION_1
External Name:

Receiver Node
Name: PT_LOCAL
Request Routing Type: Implicit
2. Request Message
Name: DELETE_USER_PROFILE
Version: VERSION_1
External Name:

In the example above, BPEL_PM will send an asynchronous message to PeopleSoft's PT_LOCAL node. The message is DELETE_USER_PROFILE.VERSION_1. Integration Broker will not send a response message. However, the Integration Broker Gateway will automatically send back an HTTP acknowledgment.

There are no transformations in this example. On your site, you may end up transforming an inbound SOAP request to a rowset-based message and transforming a rowset-based response to a non-rowset-based SOAP response. The Integration Point Wizard will walk you through all the steps. See the example below.


Integration Point Viewer - Microsoft Internet Explorer
File Edit View Favorites Tools Help

Integration Broker Point Viewer

*Sender Node:	CUST
*Transaction Type:	Synchronous
*Receiver Node:	ZIPTOCITYANDSTATE
*Request Message:	CSS_ZIP_REQUEST
*Request Message Version:	VERSION_1

View Clear

Status: Active Logging: Header and Detail

Sender Node

Name: CUST

Receiver Node

Name: ZIPTOCITYANDSTATE

Request Routing Type: Explicit

1. Request Message	4. Response Message
Name: CSS_ZIP_REQUEST	Name: CSS_ZIP_RESPONSE
Version: VERSION_1	Version: VERSION_1
External Name:	External Name:

Go to Node Transaction

Transformations

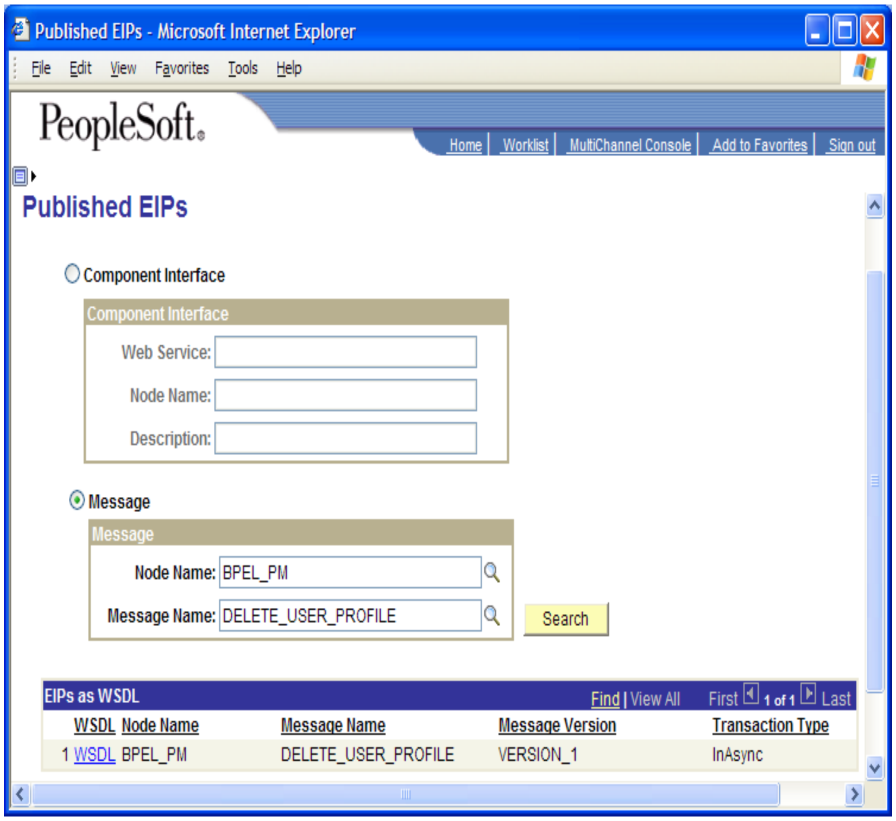
Request: CSS_ZIP_REQ Response: CSS_ZIP_RESP

Go To Transaction Modifier

2. Request Message	3. Response Message
Name: CSS_ZIP_REQUEST	Name: CSS_ZIP_RESPONSE
Version: VERSION_1	Version: VERSION_1
External Name:	External Name:

Go to Node Transaction

With the transaction created, you are now ready to generate the WSDL for the web service. The WSDL generator can be found under Integration Broker > Web Services > Published EIPs. Select Message and then specify the node that represents your BPEL system and its request message. Then click Search.


Published EIPs - Microsoft Internet Explorer
File Edit View Favorites Tools Help
PeopleSoft.
Home | Worklist | MultiChannel Console | Add to Favorites | Sign out

Published EIPs

( ) Component Interface

Component Interface

Web Service:	
Node Name:	
Description:	

(●) Message

Message

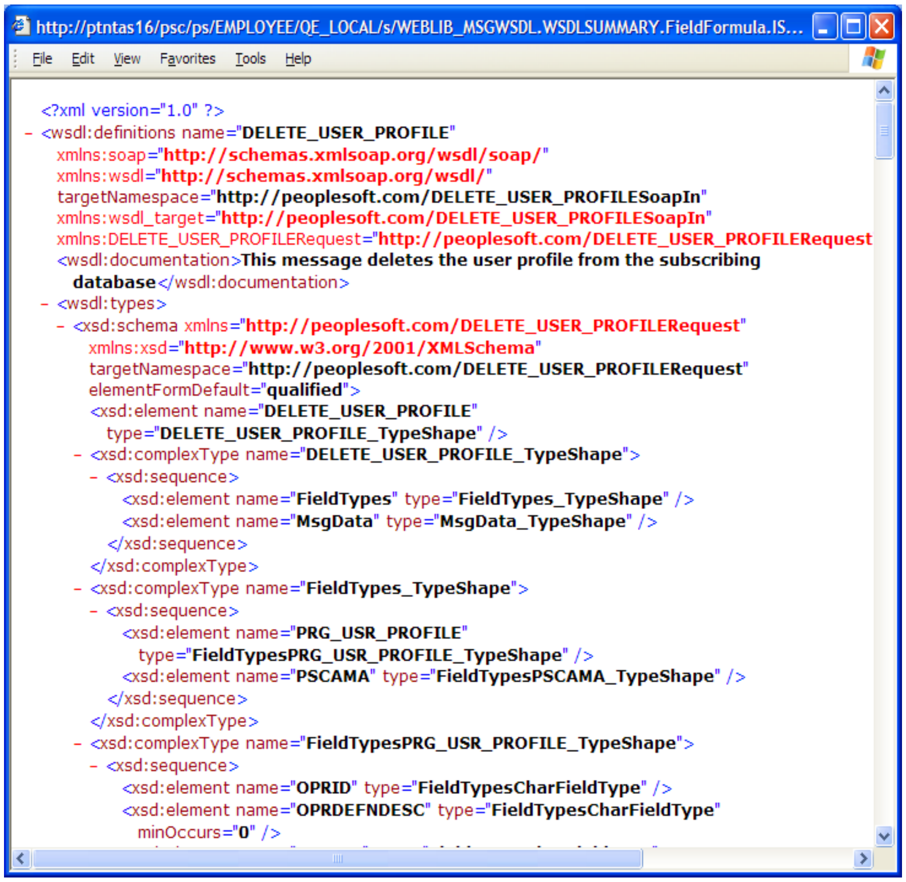
Node Name:	BPEL_PM
Message Name:	DELETE_USER_PROFILE

Search

EIPs as WSDL — Find | View All — First 1 of 1 Last

	WSDL	Node Name	Message Name	Message Version	Transaction Type
1	WSDL	BPEL_PM	DELETE_USER_PROFILE	VERSION_1	InAsync

Click the WSDL link to generate the WSDL. A new browser will open with the WSDL. You can use either just the URL in the new browser window for BPEL PM to consume the WSDL, or you can save the WSDL as a file for BPEL PM to browse.


http://ptntas16/psc/ps/EMPLOYEE/QE_LOCAL/s/WEBLIB_MSGWSDL.WSDLSUMMARY.FieldFormula.IS...
File Edit View Favorites Tools Help

```
  <?xml version="1.0" ?>
- <wsdl:definitions name="DELETE_USER_PROFILE"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    targetNamespace="http://peoplesoft.com/DELETE_USER_PROFILESoapIn"
    xmlns:wsdl_target="http://peoplesoft.com/DELETE_USER_PROFILESoapIn"
    xmlns:DELETE_USER_PROFILERequest="http://peoplesoft.com/DELETE_USER_PROFILERequest
    <wsdl:documentation>This message deletes the user profile from the subscribing
      database</wsdl:documentation>
  - <wsdl:types>
    - <xsd:schema xmlns="http://peoplesoft.com/DELETE_USER_PROFILERequest"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://peoplesoft.com/DELETE_USER_PROFILERequest"
        elementFormDefault="qualified">
        <xsd:element name="DELETE_USER_PROFILE"
          type="DELETE_USER_PROFILE_TypeShape" />
      - <xsd:complexType name="DELETE_USER_PROFILE_TypeShape">
        - <xsd:sequence>
            <xsd:element name="FieldTypes" type="FieldTypes_TypeShape" />
            <xsd:element name="MsgData" type="MsgData_TypeShape" />
          </xsd:sequence>
        </xsd:complexType>
      - <xsd:complexType name="FieldTypes_TypeShape">
        - <xsd:sequence>
            <xsd:element name="PRG_USR_PROFILE"
              type="FieldTypesPRG_USR_PROFILE_TypeShape" />
            <xsd:element name="PSCAMA" type="FieldTypesPSCAMA_TypeShape" />
          </xsd:sequence>
        </xsd:complexType>
      - <xsd:complexType name="FieldTypesPRG_USR_PROFILE_TypeShape">
        - <xsd:sequence>
            <xsd:element name="OPRID" type="FieldTypesCharFieldType" />
            <xsd:element name="OPRDEFNDESC" type="FieldTypesCharFieldType"
              minOccurs="0" />
```

The WSDL is now ready to be consumed by BPEL PM.

PeopleSoft consuming BPEL

PeopleSoft Consuming Web Services and Initiating a BPEL PM Process

In order for a PeopleSoft application to initiate a BPEL PM process, either the PeopleSoft application must consume a BPEL PM provided web service or BPEL PM must subscribe to a PeopleSoft notification message. This section describes how to consume a BPEL PM (or any external 3rd party) web service.

You can consume both synchronous and asynchronous web services using the Import WSDL wizard. The wizard guides you in creating the metadata required to invoke the web service (Node definition, messages and Transaction).

After the wizard creates the metadata, you must either write an XSLT transformation to transform a rowset-based (structured) message to the BPEL PM SOAP request or notification format, or you must write a PeopleCode program using the XMLDoc and/or SOAPDoc classes to populate the SOAP request/notification. You can create the XSLT transformation graphically using the XSLT graphical mapper that comes with BPEL PM.

Rowset-Based or Non-Rowset-Based Messages: Which Should I Use? When consuming an external web service, you can code to it directly using XMLDoc PeopleCode (non-rowset-based) or you can transform a delivered (or custom) rowset-based message to the required SOAP format. If there is a delivered message in place that is similar in nature to the web service you are consuming, then use it (along with a transformation program, if necessary). When the payload for the SOAP request to BPEL PM comes from more than one scroll, it will be easier to use the CopyRowset PeopleCode functionality to populate rowset-based messages and transform them using XSLT into the required SOAP format.

When the payload will be just a few fields from a single scroll level, you can use either rowset-based or non-rowset based messages. For a POC or rapid development, coding XMLDoc PeopleCode to a non-rowset-based message may take very little effort. However, if the message needs to be sent from more than one place or if there is a chance that the message format will ever change, or just to be more production worthy, the XMLDoc PeopleCode should be placed in an Application Class, where it can be called from any Component. This way, you will need to update the XMLDoc PeopleCode in just one place, instead of in several places. If you use rowset-based messages with transformations to the BPEL PM requested format, should the format ever change, the only objects you will ever need to update are the XSLT programs.

High level view of the steps to consume a synchronous BPEL PM web service:

- Import the WSDL into PeopleSoft using the WSDL Import Wizard.
- Use the wizard to create the metadata for each service operation you want to invoke.
- If you are using XMLDoc PeopleCode to invoke the service, add the code to an app class and/or a component and you are done. If you are using XSLT to transform from structured messages to unstructured messages, continue on to the following steps:
- Create the XSLT for the outbound request. A graphical mapper is included with BPEL PM if you do not wan to write the code manually.
- Create the XSLT for the inbound response.
- Create an IB outbound transaction on the BPEL (3rd party node) for the structured messages.
- Create the IB Relationship and Transaction Modifier to apply the XSLT to transform structured messages to/from SOAP messages.
- Optional: Use the Integration Point Viewer to review the configuration.

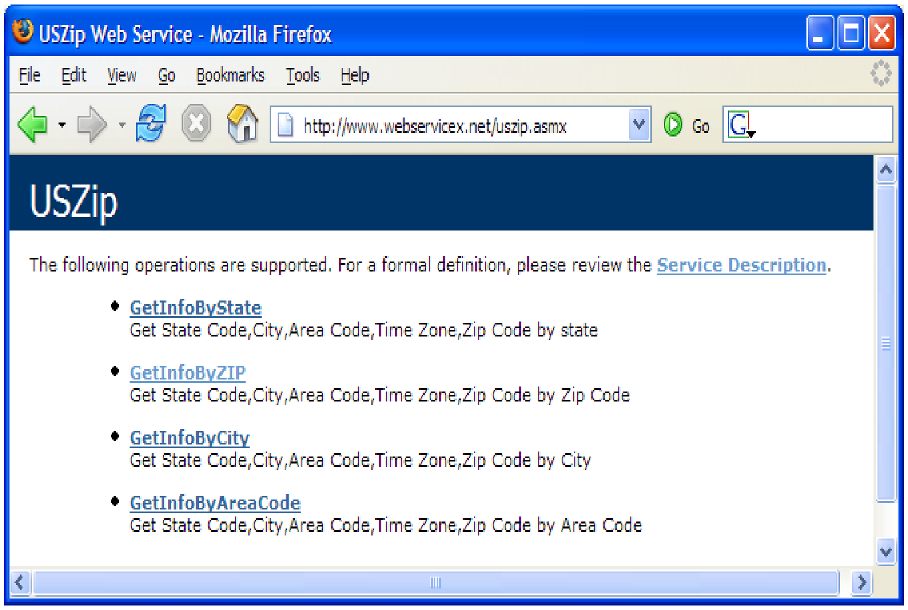
High level view of the steps to consume an asynchronous BPEL PM web service:

- Import the WSDL into PeopleSoft using the WSDL Import Wizard.
- Use the wizard to create the metadata for each service operation you want to invoke.
- If you are using XMLDoc PeopleCode to invoke the service, add the code to an app class and/or a component and you are done. If you are using XSLT to transform from a structured message to an unstructured message, continue on to the following steps:
- Create the XSLT for the outbound message. A graphical mapper is included with BPEL PM if you do not wan to write the code manually.

- If BPEL PM is going to send a response you can create XSLT for the inbound response. (You can also mix and match, i.e. transform the request but don't transform the response.)
- Create an IB outbound transaction on the BPEL (3rd party node) for the structured message.
- Create the IB Relationship and Transaction Modifier to apply the XSLT to transform structured message to/from SOAP or the appropriate format.
- Optional: Use the Integration Point Viewer to review the configuration.

Consuming a BPEL PM or 3rd Party Web Service in PeopleSoft:

For this section, we will consume a synchronous web service that is readily available on the internet. The principles and practices are the same whether the web service comes from BPEL PM or any other provider. Identify the service to be consumed. In this example we are using the GetInfoByZip zip code lookup service from www.webservicex.net/uszip.asmx.


USZip Web Service - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://www.webservicex.net/uszip.asmx
Go

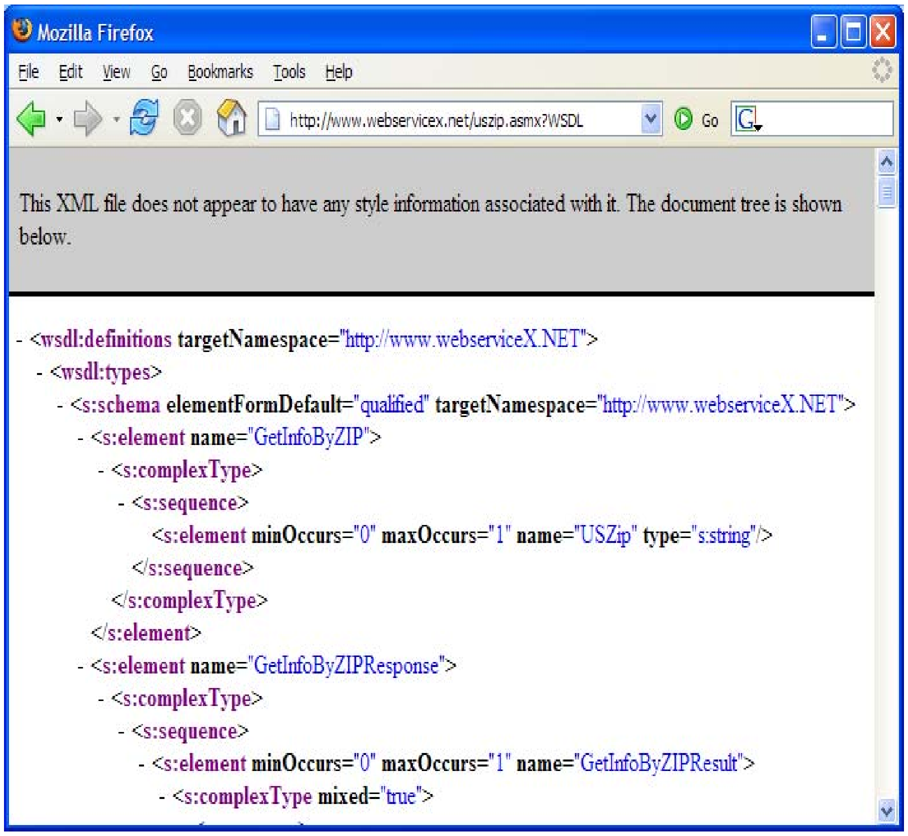
USZip

The following operations are supported. For a formal definition, please review the Service Description.

- GetInfoByState
  Get State Code,City,Area Code,Time Zone,Zip Code by state
- GetInfoByZIP
  Get State Code,City,Area Code,Time Zone,Zip Code by Zip Code
- GetInfoByCity
  Get State Code,City,Area Code,Time Zone,Zip Code by City
- GetInfoByAreaCode
  Get State Code,City,Area Code,Time Zone,Zip Code by Area Code

Find the WSDL for the service you want to consume. The WSDL for this service is at www.webservicex.net/uszip?WSDL

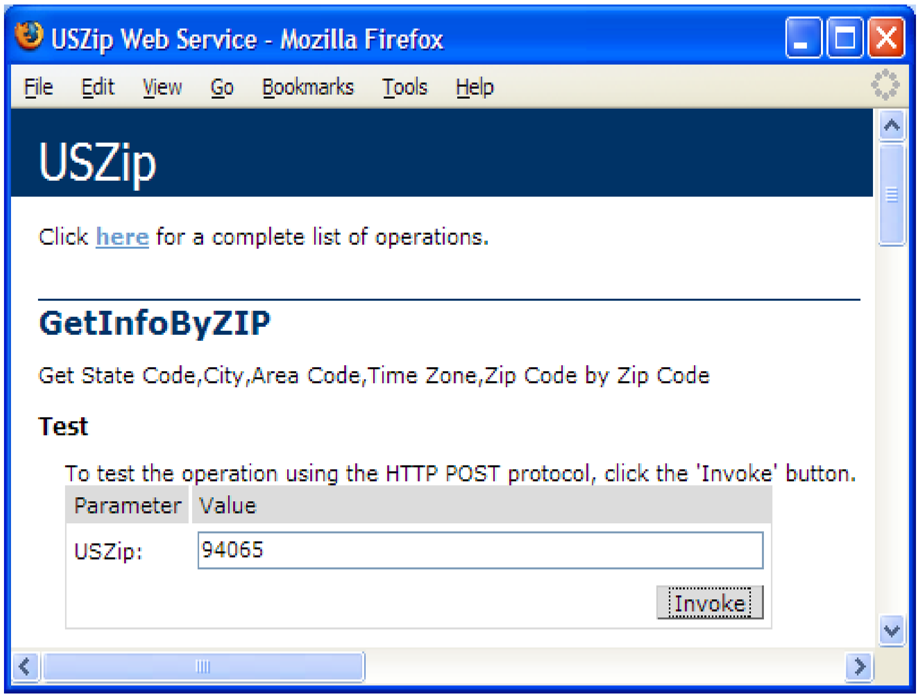
.


Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://www.webservicex.net/uszip.asmx?WSDL
Go

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
- <wsdl:definitions targetNamespace="http://www.webserviceX.NET">
  - <wsdl:types>
    - <s:schema elementFormDefault="qualified" targetNamespace="http://www.webserviceX.NET">
      - <s:element name="GetInfoByZIP">
        - <s:complexType>
          - <s:sequence>
              <s:element minOccurs="0" maxOccurs="1" name="USZip" type="s:string"/>
            </s:sequence>
          </s:complexType>
        </s:element>
      - <s:element name="GetInfoByZIPResponse">
        - <s:complexType>
          - <s:sequence>
            - <s:element minOccurs="0" maxOccurs="1" name="GetInfoByZIPResult">
              - <s:complexType mixed="true">
```

Optional: If the service provider provides a test page, test the web service to understand how it behaves before you consume and code to it.


USZip Web Service - Mozilla Firefox
File Edit View Go Bookmarks Tools Help

USZip

Click here for a complete list of operations.

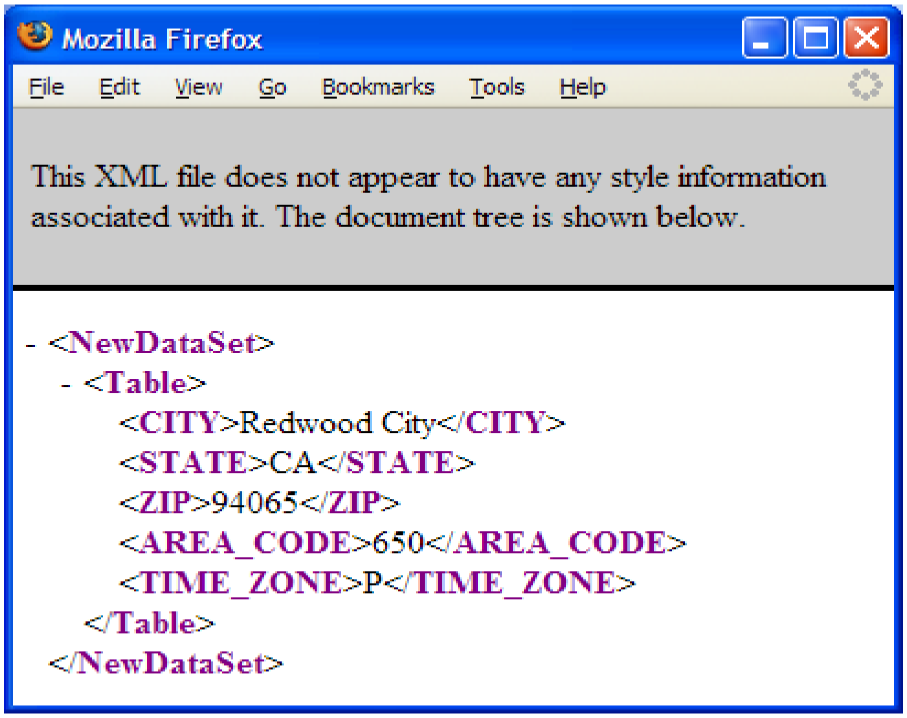
GetInfoByZIP

Get State Code,City,Area Code,Time Zone,Zip Code by Zip Code

Test

To test the operation using the HTTP POST protocol, click the 'Invoke' button.

Parameter	Value
USZip:	94065



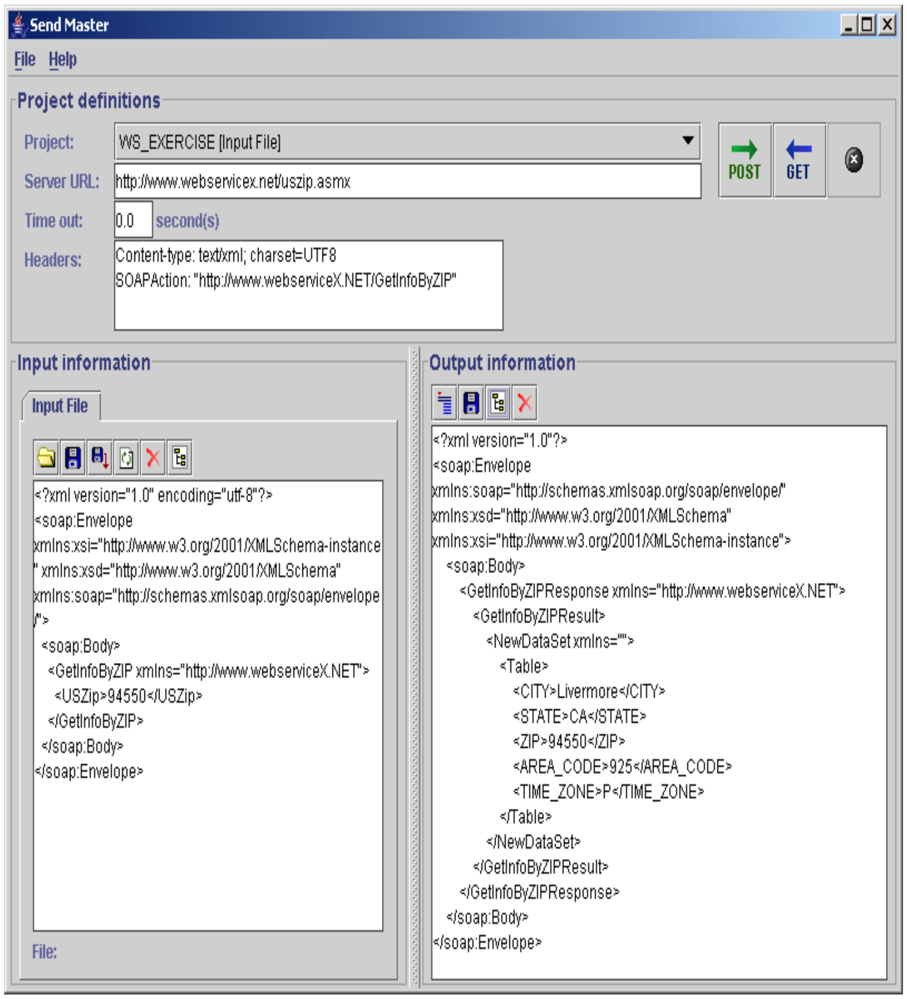
This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
- <NewDataSet>
  - <Table>
      <CITY>Redwood City</CITY>
      <STATE>CA</STATE>
      <ZIP>94065</ZIP>
      <AREA_CODE>650</AREA_CODE>
      <TIME_ZONE>P</TIME_ZONE>
    </Table>
  </NewDataSet>
```

Optional but strongly recommended: Whether or not the provider provides a test page, you can and should test the web service with the PeopleSoft provided tool, SendMaster. SendMaster is a stand-alone test tool that is fully documented in the Integration Broker PeopleBooks.

In the example below, simply copy and past the URL and header information from the test page in to the appropriate fields, then copy and past the sample SOAP request into the Input Information Section. Finally, replace “string” in the SOAP message with your actual zip code. Then click POST. If you don’t get a valid SOAP response, either you have connectivity issues or the provider is unavailable.

The bottom line is that if SendMaster can’t successfully invoke the web service, then don’t waste your time trying to get PeopleSoft to do it.

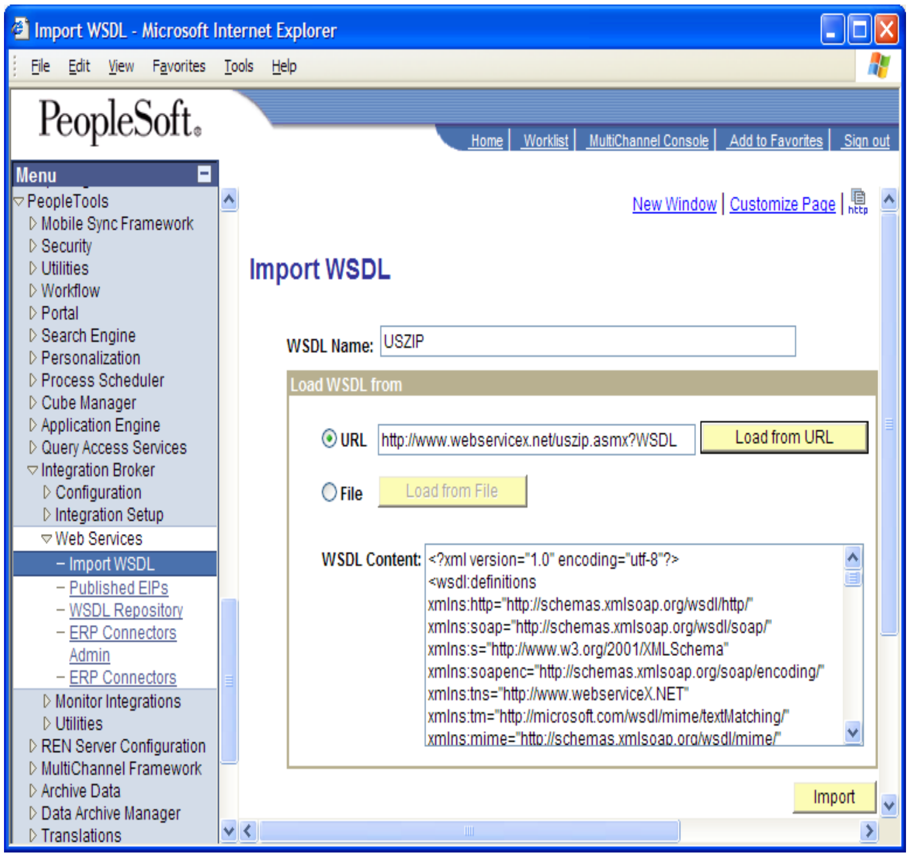


Project: WS_EXERCISE [Input File]
Server URL: http://www.webservicex.net/uszip.asmx
Time out: 0.0 second(s)
Headers:
```
Content-type: text/xml; charset=UTF8
SOAPAction: "http://www.webserviceX.NET/GetInfoByZIP"
```

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance
" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope
/">
  <soap:Body>
    <GetInfoByZIP xmlns="http://www.webserviceX.NET">
      <USZip>94550</USZip>
    </GetInfoByZIP>
  </soap:Body>
</soap:Envelope>
```
File:

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soap:Body>
    <GetInfoByZIPResponse xmlns="http://www.webserviceX.NET">
      <GetInfoByZIPResult>
        <NewDataSet xmlns="">
          <Table>
            <CITY>Livermore</CITY>
            <STATE>CA</STATE>
            <ZIP>94550</ZIP>
            <AREA_CODE>925</AREA_CODE>
            <TIME_ZONE>P</TIME_ZONE>
          </Table>
        </NewDataSet>
      </GetInfoByZIPResult>
    </GetInfoByZIPResponse>
  </soap:Body>
</soap:Envelope>
```

Navigate to the WSDL Import Wizard to consume the web service. The name you choose to assign the WSDL is arbitrary. After you cut and paste the WSDL URL into the wizard, click the "Load from URL" button. When the WSDL shows in the WSDL Content field, click the "Import" button.



WSDL Name: USZIP

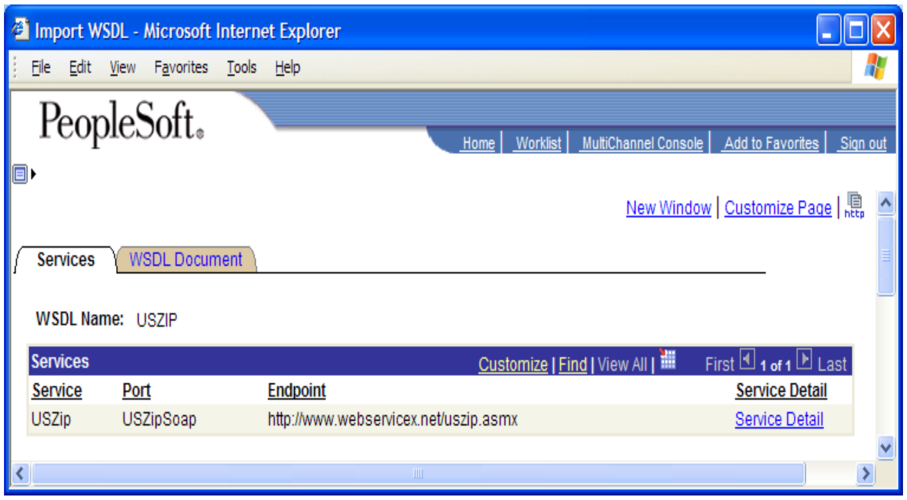
URL: http://www.webservicex.net/uszip.asmx?WSDL

File

WSDL Content:

```
<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tns="http://www.webserviceX.NET"
xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
```

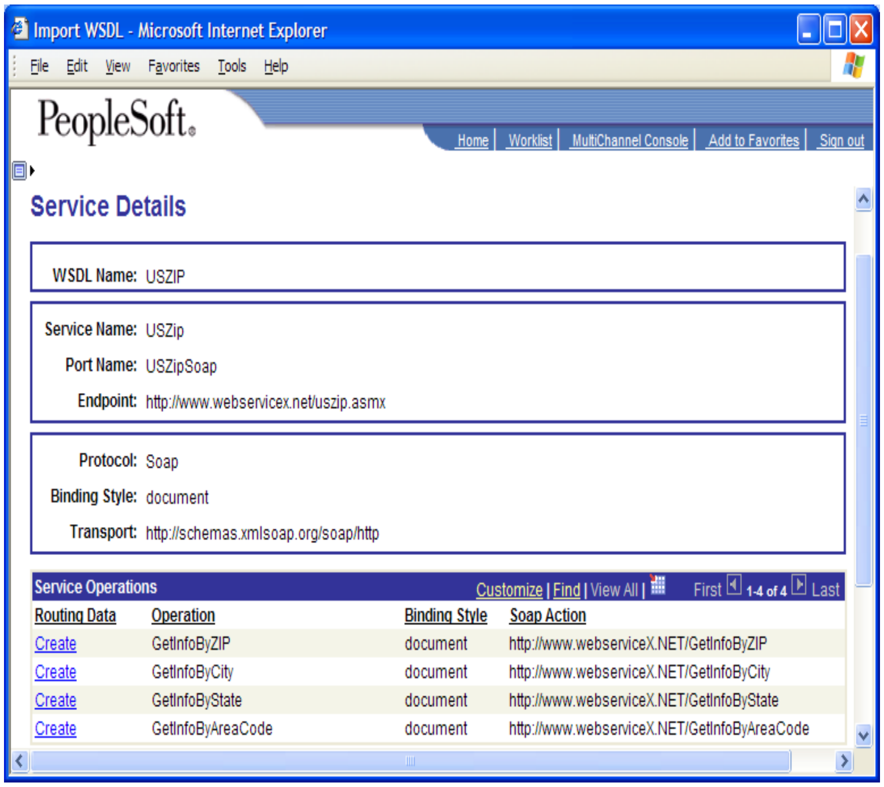
Once the WSDL is imported, click the “Service Detail” link.


Import WSDL - Microsoft Internet Explorer
File Edit View Favorites Tools Help
PeopleSoft.
Home | Worklist | MultiChannel Console | Add to Favorites | Sign out
New Window | Customize Page
Services
WSDL Document
Customize | Find | View All |
First 1 of 1 Last

WSDL Name: USZIP

Services

Service	Port	Endpoint	Service Detail
USZip	USZipSoap	http://www.webservicex.net/uszip.asmx	Service Detail


Import WSDL - Microsoft Internet Explorer
File Edit View Favorites Tools Help
PeopleSoft.
Home | Worklist | MultiChannel Console | Add to Favorites | Sign out

Service Details

WSDL Name: USZIP

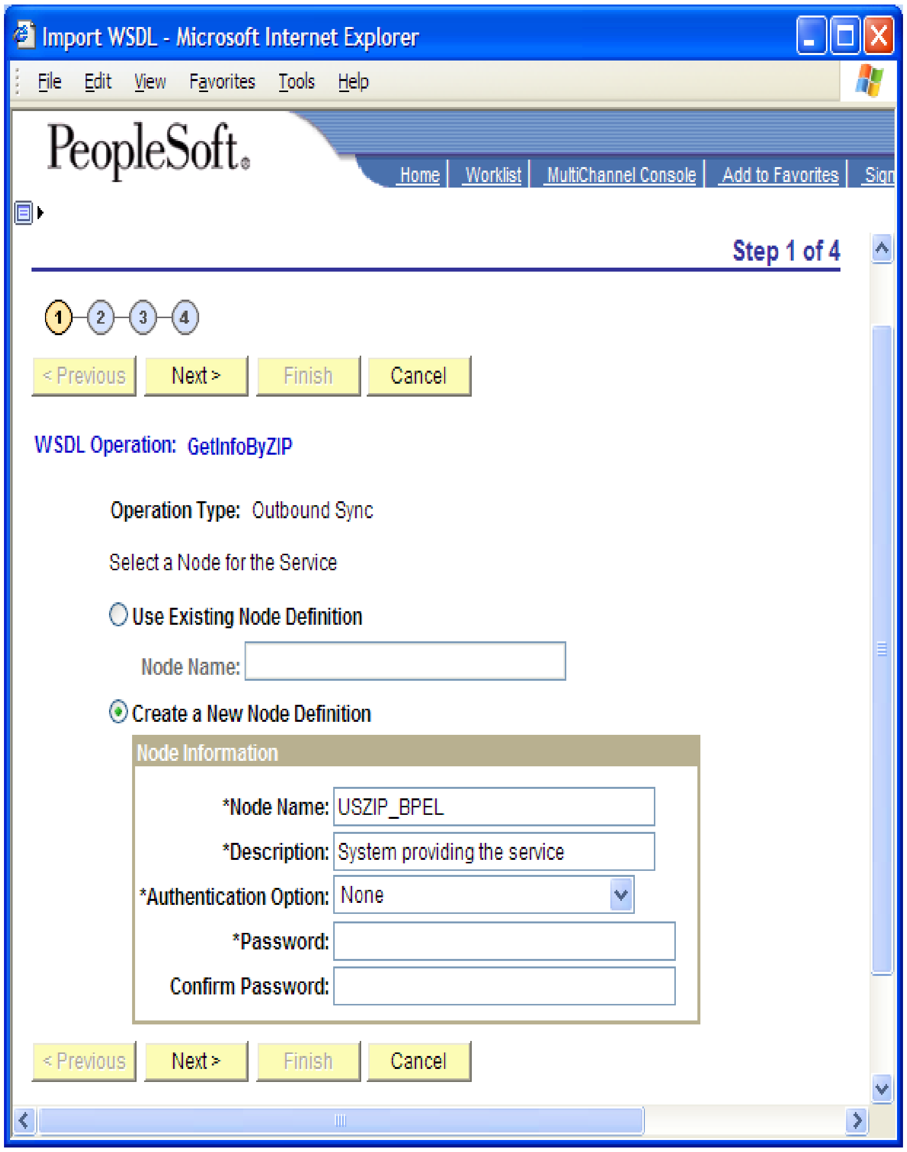
Service Name: USZip
Port Name: USZipSoap
Endpoint: http://www.webservicex.net/uszip.asmx

Protocol: Soap
Binding Style: document
Transport: http://schemas.xmlsoap.org/soap/http

Service Operations — Customize | Find | View All | First 1-4 of 4 Last

Routing Data	Operation	Binding Style	Soap Action
Create	GetInfoByZIP	document	http://www.webserviceX.NET/GetInfoByZIP
Create	GetInfoByCity	document	http://www.webserviceX.NET/GetInfoByCity
Create	GetInfoByState	document	http://www.webserviceX.NET/GetInfoByState
Create	GetInfoByAreaCode	document	http://www.webserviceX.NET/GetInfoByAreaCode

The “Create” links allow you to create the integration metadata to invoke any of the operations. Click the first “Create” link for GetInfoByZip to launch the wizard that creates the metadata.


Import WSDL - Microsoft Internet Explorer
File Edit View Favorites Tools Help
PeopleSoft.
Home | Worklist | MultiChannel Console | Add to Favorites | Sign
Step 1 of 4
1 2 3 4
< Previous Next > Finish Cancel

WSDL Operation: GetInfoByZIP

Operation Type: Outbound Sync

Select a Node for the Service

Use Existing Node Definition [✓ Create a New Node Definition]

Node Name:

Node Information

*Node Name: USZIP_BPEL

*Description: System providing the service

*Authentication Option: None

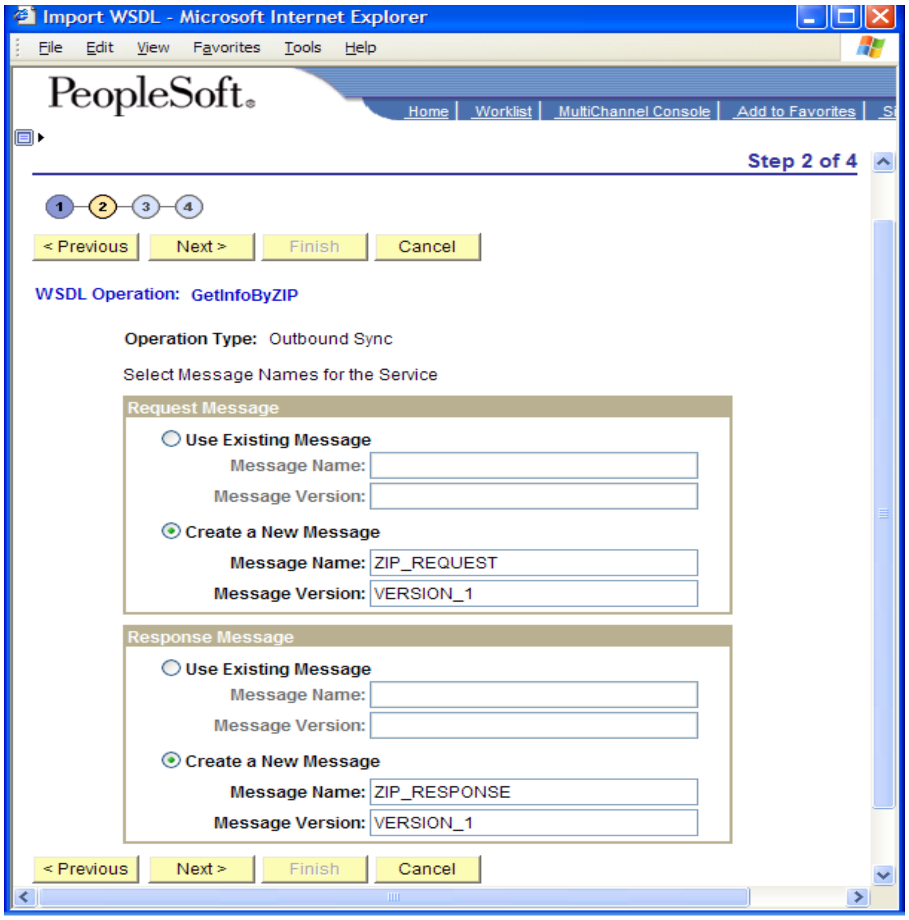
*Password:

Confirm Password:

< Previous Next > Finish Cancel

If you have consumed a web service from the BPEL/3rd party system before, then Use Existing Node Definition and specify the node name. If this is your first time importing WSDL from (or integrating with) the BPEL/3rd party system then Create a New Node Definition. You can choose Authentication Option: None and add authentication, such as SSL, later.

Click Next and specify the names of the request and response messages for consuming the service. These are non-rowset-based (unstructured) messages. If you choose Create a New Message, the wizard will create the messages in the application designer for you. Click Next.


Import WSDL - Microsoft Internet Explorer
File Edit View Favorites Tools Help
PeopleSoft.
Home | Worklist | MultiChannel Console | Add to Favorites | Si
Step 2 of 4
1 2 3 4
< Previous
Next >
Finish
Cancel

WSDL Operation: GetInfoByZIP

Operation Type: Outbound Sync

Select Message Names for the Service

Request Message

( ) Use Existing Message
Message Name:
Message Version:

(•) Create a New Message
Message Name: ZIP_REQUEST
Message Version: VERSION_1

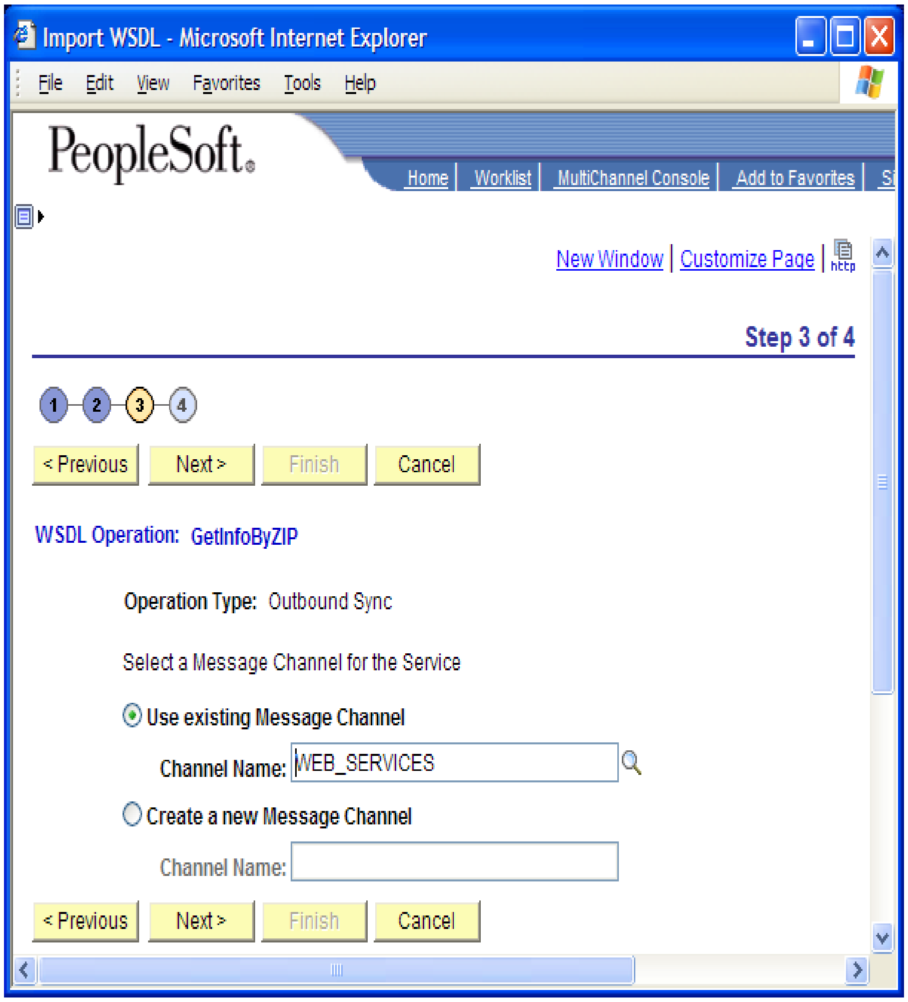
Response Message

( ) Use Existing Message
Message Name:
Message Version:

(•) Create a New Message
Message Name: ZIP_RESPONSE
Message Version: VERSION_1

Channels, for synchronous messages only, are arbitrary. You can choose any channel you want but a generic channel for all synchronous web services makes the most sense. For asynchronous services, however, choosing the right channel for partitioning (and performance) is critical. Asynchronous messages should be grouped into channels with other messages that are relevant to the business process and partitioned by common keys.

In the example below, we are specifying the web services channel because it is a synchronous web service. After you specify the channel, click Next.


Import WSDL - Microsoft Internet Explorer
File Edit View Favorites Tools Help
PeopleSoft.
Home | Worklist | MultiChannel Console | Add to Favorites | Si
New Window | Customize Page | http
1 2 3 4
< Previous Next > Finish Cancel

Step 3 of 4

WSDL Operation: GetInfoByZIP

Operation Type: Outbound Sync

Select a Message Channel for the Service

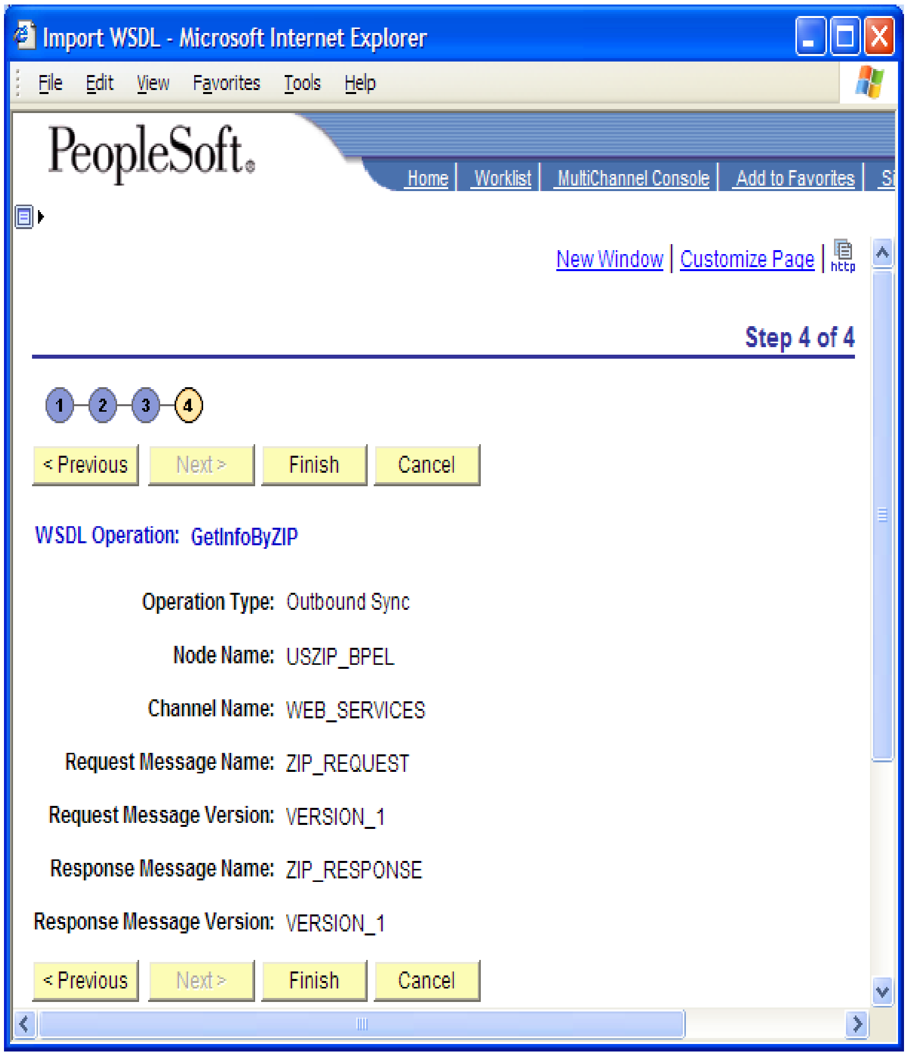
[✓ Use existing Message Channel]

Channel Name: WEB_SERVICES

Create a new Message Channel

Channel Name:

If the summary information looks correct, click Finish.

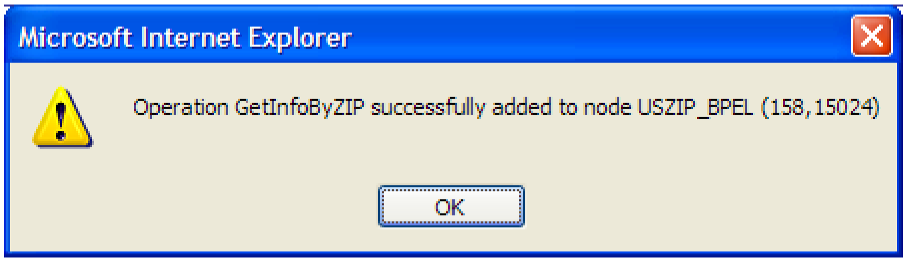

Import WSDL - Microsoft Internet Explorer
File Edit View Favorites Tools Help
PeopleSoft.
Home | Worklist | MultiChannel Console | Add to Favorites | Si
New Window | Customize Page | http
1 2 3 4
< Previous Next > Finish Cancel

Step 4 of 4

WSDL Operation: GetInfoByZIP

Operation Type:	Outbound Sync
Node Name:	USZIP_BPEL
Channel Name:	WEB_SERVICES
Request Message Name:	ZIP_REQUEST
Request Message Version:	VERSION_1
Response Message Name:	ZIP_RESPONSE
Response Message Version:	VERSION_1

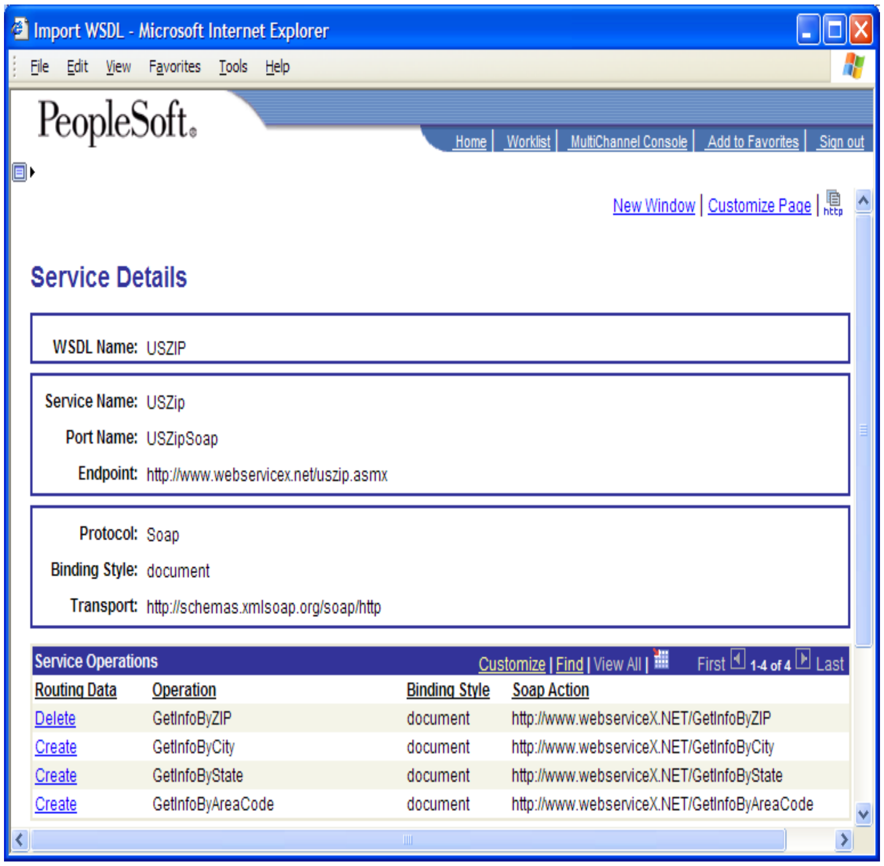
Upon successful creation of the integration metadata, the following is displayed:



Operation GetInfoByZIP successfully added to node USZIP_BPEL (158,15024)

Notice that the Create link has been replaced with a delete link for deleting the integration metadata.

Note, if you want to delete and recreate the integration metadata for invoking the web service operation, you must delete the transaction by clicking this Delete link. Deleting the transaction from the Node Definition page is not enough.


Import WSDL - Microsoft Internet Explorer
File Edit View Favorites Tools Help
PeopleSoft.
Home | Worklist | MultiChannel Console | Add to Favorites | Sign out
New Window | Customize Page

Service Details

WSDL Name: USZIP

Service Name: USZip
Port Name: USZipSoap
Endpoint: http://www.webservicex.net/uszip.asmx

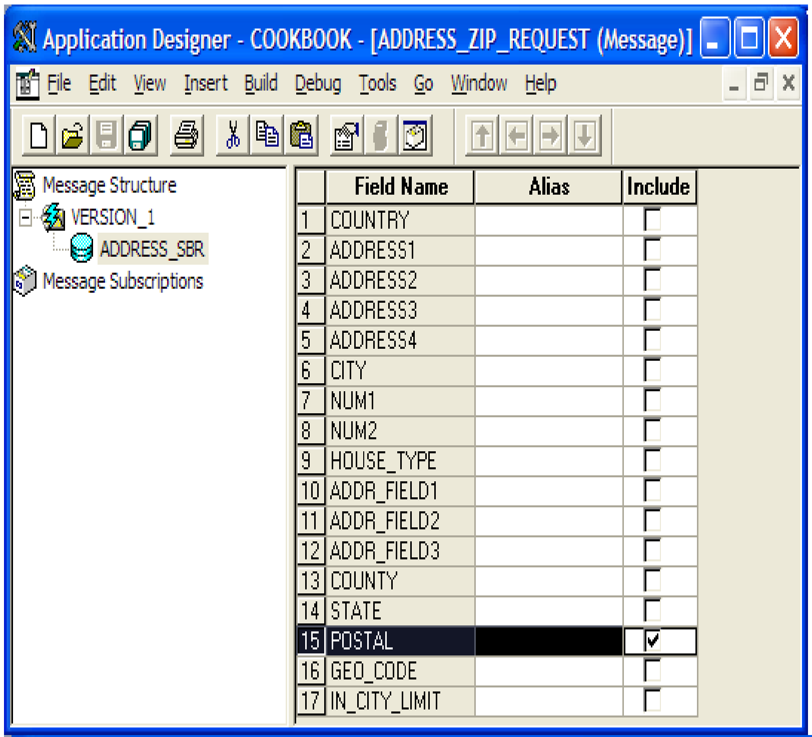
Protocol: Soap
Binding Style: document
Transport: http://schemas.xmlsoap.org/soap/http

Service Operations — Customize | Find | View All | First 1-4 of 4 Last

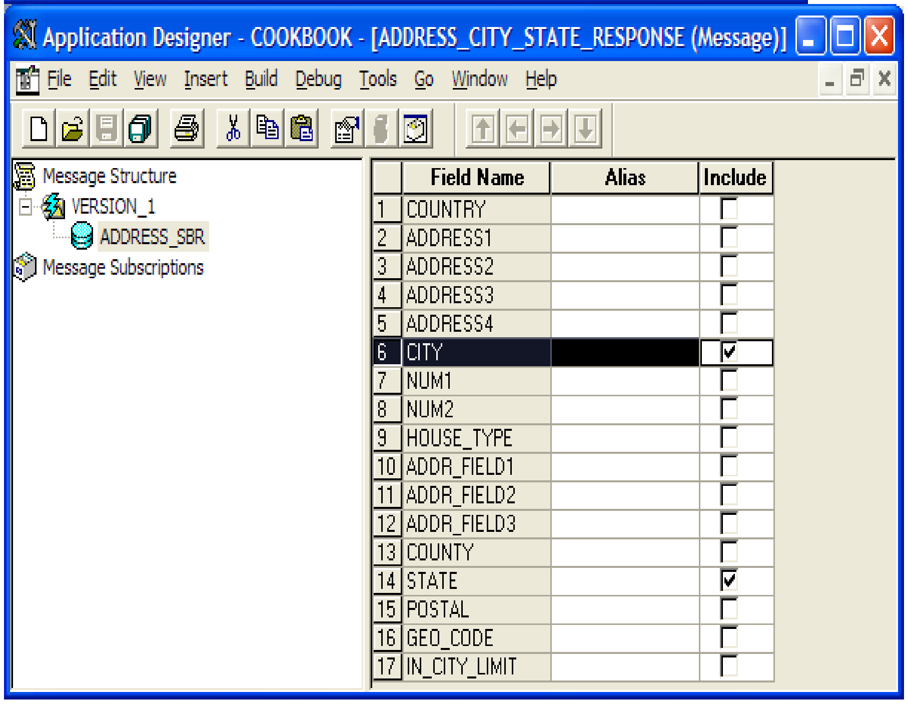
Routing Data	Operation	Binding Style	Soap Action
Delete	GetInfoByZIP	document	http://www.webserviceX.NET/GetInfoByZIP
Create	GetInfoByCity	document	http://www.webserviceX.NET/GetInfoByCity
Create	GetInfoByState	document	http://www.webserviceX.NET/GetInfoByState
Create	GetInfoByAreaCode	document	http://www.webserviceX.NET/GetInfoByAreaCode

If you will be invoking the BPEL or 3rd party web service using XMLDoc PeopleCode, you are now ready to write the code and insert it in the App Class or PeopleCode event that will call it. If you will be invoking the web service by transforming structured messages into unstructured messages, continue following the rest of the steps in this section.

Before we use the Integration Point Wizard to create the remaining integration metadata, we must first verify the existence of, or create, the structured messages and the transformation programs. Create the following two structured messages by dragging or inserting the ADDRESS_SBR record into the VERSION_1 folder of the message. Notice that in the ADDRESS_ZIP_REQUEST message, only the POSTAL field is checked in the Include column. Also notice that in the ADDRESS_CITY_STATE_RESPONSE message has only the City and State fields checked in the Include column.

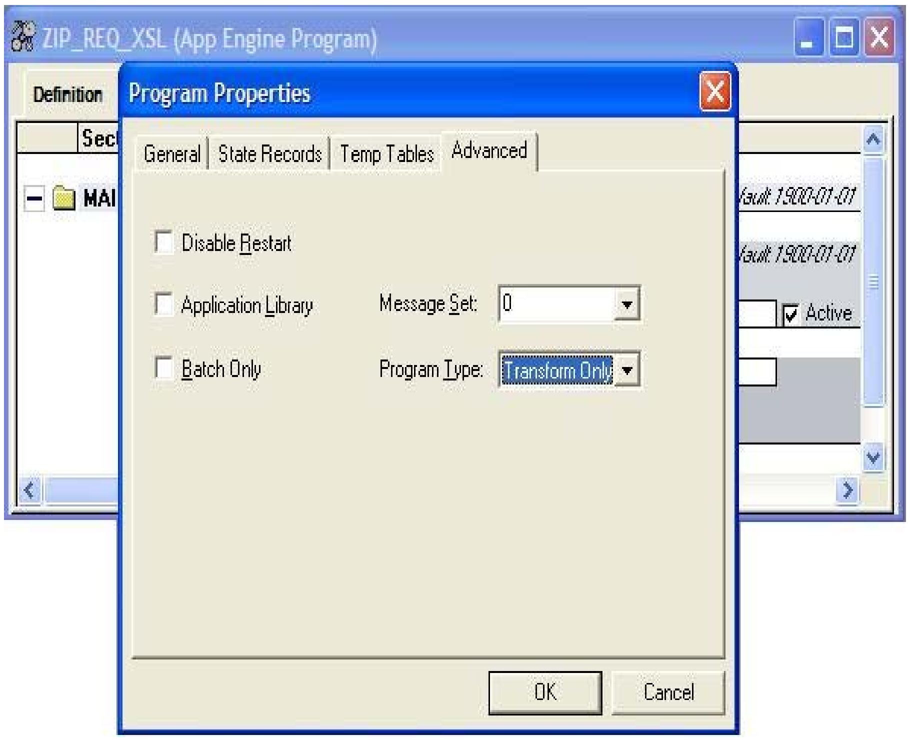

Application Designer - COOKBOOK - [ADDRESS_ZIP_REQUEST (Message)]
File Edit View Insert Build Debug Tools Go Window Help
Message Structure
VERSION_1
ADDRESS_SBR
Message Subscriptions

	Field Name	Alias	Include
1	COUNTRY		☐
2	ADDRESS1		☐
3	ADDRESS2		☐
4	ADDRESS3		☐
5	ADDRESS4		☐
6	CITY		☐
7	NUM1		☐
8	NUM2		☐
9	HOUSE_TYPE		☐
10	ADDR_FIELD1		☐
11	ADDR_FIELD2		☐
12	ADDR_FIELD3		☐
13	COUNTY		☐
14	STATE		☐
15	POSTAL		✓
16	GEO_CODE		☐
17	IN_CITY_LIMIT		☐


Application Designer - COOKBOOK - [ADDRESS_CITY_STATE_RESPONSE (Message)]
File Edit View Insert Build Debug Tools Go Window Help
Message Structure
VERSION_1
ADDRESS_SBR
Message Subscriptions

	Field Name	Alias	Include
1	COUNTRY		☐
2	ADDRESS1		☐
3	ADDRESS2		☐
4	ADDRESS3		☐
5	ADDRESS4		☐
6	CITY		✓
7	NUM1		☐
8	NUM2		☐
9	HOUSE_TYPE		☐
10	ADDR_FIELD1		☐
11	ADDR_FIELD2		☐
12	ADDR_FIELD3		☐
13	COUNTY		☐
14	STATE		✓
15	POSTAL		☐
16	GEO_CODE		☐
17	IN_CITY_LIMIT		☐

Create the ZIP_REQ_XSL Application Engine programs that will house the request transformation XSLT code. Make sure to specify Transform Only in the Properties Page.


ZIP_REQ_XSL (App Engine Program)

Program Properties

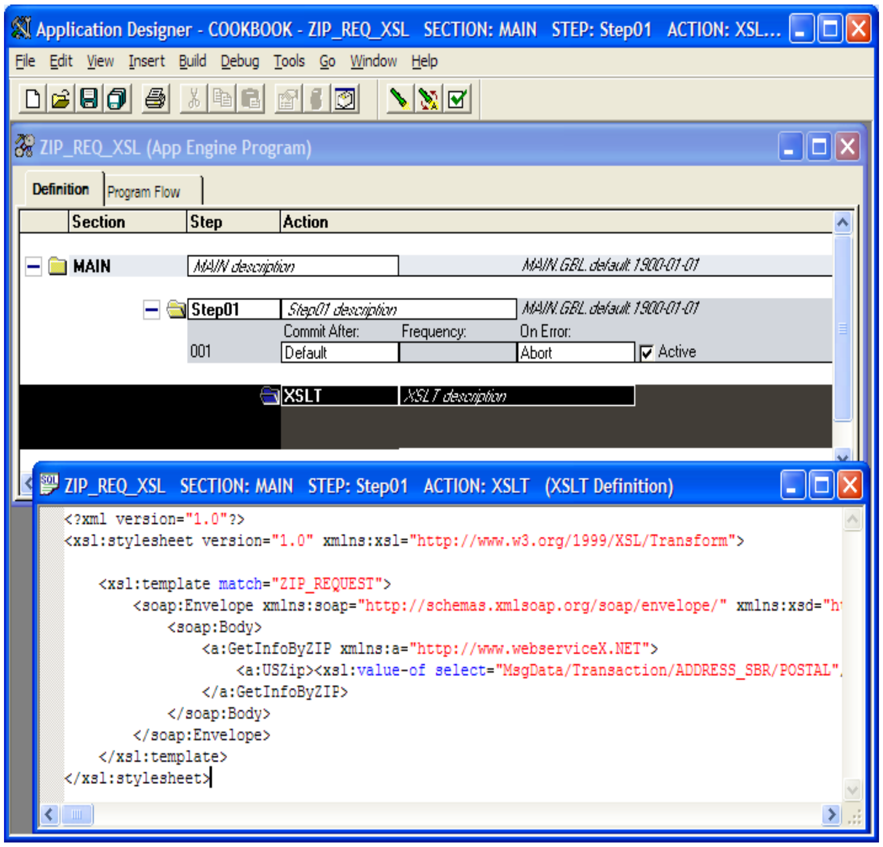
General | State Records | Temp Tables | Advanced

- [ ] Disable Restart
- [ ] Application Library
- [ ] Batch Only

Message Set: 0

Program Type: Transform Only

The resulting Application Engine program should look like the following:

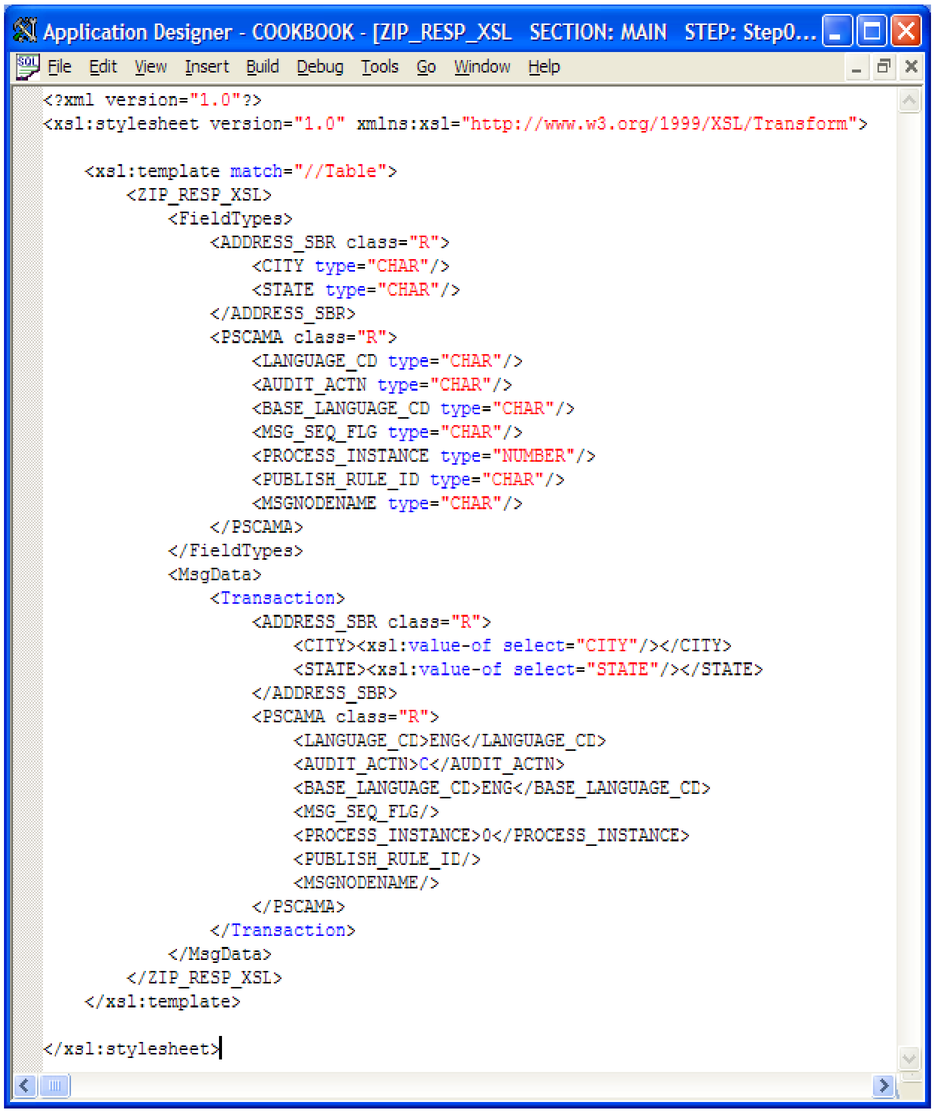

Application Designer - COOKBOOK - ZIP_REQ_XSL SECTION: MAIN STEP: Step01 ACTION: XSL...
File Edit View Insert Build Debug Tools Go Window Help
ZIP_REQ_XSL (App Engine Program)
Definition
Program Flow
Section
Step
Action
MAIN
MAIN description
MAIN.GBL.default.1900-01-01
Step01
Step01 description
MAIN.GBL.default.1900-01-01
Commit After:
Frequency:
On Error:
001
Default
Abort
Active
XSLT
XSLT description
ZIP_REQ_XSL SECTION: MAIN STEP: Step01 ACTION: XSLT (XSLT Definition)

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

    <xsl:template match="ZIP_REQUEST">
        <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="h
            <soap:Body>
                <a:GetInfoByZIP xmlns:a="http://www.webserviceX.NET">
                    <a:USZip><xsl:value-of select="MsgData/Transaction/ADDRESS_SBR/POSTAL",
                </a:GetInfoByZIP>
            </soap:Body>
        </soap:Envelope>
    </xsl:template>
</xsl:stylesheet>
```

Here is sample XSLT code that you can use:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

   <xsl:template match="ZIP_REQUEST">
      <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
         <soap:Body>
            <a:GetInfoByZIP xmlns:a="http://www.webserviceX.NET">
               <a:USZip><xsl:value-of
select="MsgData/Transaction/ADDRESS_SBR/POSTAL"/></a:USZip>
            </a:GetInfoByZIP>
         </soap:Body>
      </soap:Envelope>
   </xsl:template>
</xsl:stylesheet>
```

Create the ZIP_RESP_XSL Application Engine programs that will house the response transformation XSLT code. Make sure to specify Transform Only in the Properties Page.


Application Designer - COOKBOOK - [ZIP_RESP_XSL SECTION: MAIN STEP: Step0...
File Edit View Insert Build Debug Tools Go Window Help

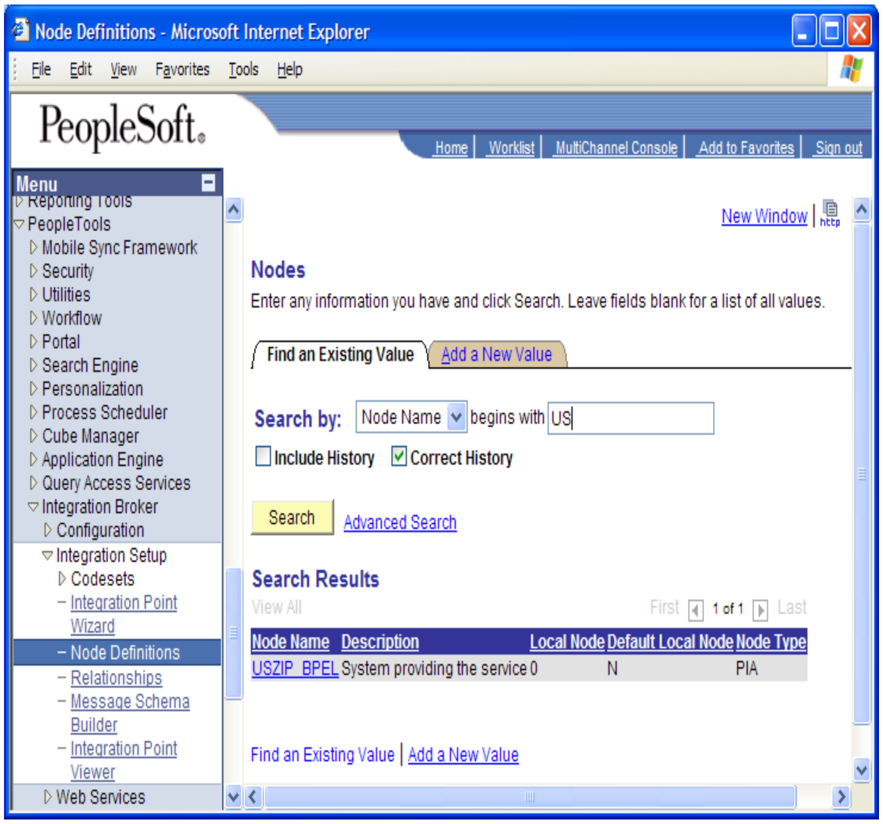
```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

    <xsl:template match="//Table">
        <ZIP_RESP_XSL>
            <FieldTypes>
                <ADDRESS_SBR class="R">
                    <CITY type="CHAR"/>
                    <STATE type="CHAR"/>
                </ADDRESS_SBR>
                <PSCAMA class="R">
                    <LANGUAGE_CD type="CHAR"/>
                    <AUDIT_ACTN type="CHAR"/>
                    <BASE_LANGUAGE_CD type="CHAR"/>
                    <MSG_SEQ_FLG type="CHAR"/>
                    <PROCESS_INSTANCE type="NUMBER"/>
                    <PUBLISH_RULE_ID type="CHAR"/>
                    <MSGNODENAME type="CHAR"/>
                </PSCAMA>
            </FieldTypes>
            <MsgData>
                <Transaction>
                    <ADDRESS_SBR class="R">
                        <CITY><xsl:value-of select="CITY"/></CITY>
                        <STATE><xsl:value-of select="STATE"/></STATE>
                    </ADDRESS_SBR>
                    <PSCAMA class="R">
                        <LANGUAGE_CD>ENG</LANGUAGE_CD>
                        <AUDIT_ACTN>C</AUDIT_ACTN>
                        <BASE_LANGUAGE_CD>ENG</BASE_LANGUAGE_CD>
                        <MSG_SEQ_FLG/>
                        <PROCESS_INSTANCE>0</PROCESS_INSTANCE>
                        <PUBLISH_RULE_ID/>
                        <MSGNODENAME/>
                    </PSCAMA>
                </Transaction>
            </MsgData>
        </ZIP_RESP_XSL>
    </xsl:template>

</xsl:stylesheet>
```

Here is sample XSLT code that you can use. If you don't know how to write XSLT code, you can use the graphical XSLT mapper in JDeveloper and have it write the code for you

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

   <xsl:template match="//Table">
      <ZIP_RESP_XSL>
         <FieldTypes>
            <ADDRESS_SBR class="R">
               <CITY type="CHAR"/>
```

```
                <STATE type="CHAR"/>
            </ADDRESS_SBR>
            <PSCAMA class="R">
                <LANGUAGE_CD type="CHAR"/>
                <AUDIT_ACTN type="CHAR"/>
                <BASE_LANGUAGE_CD type="CHAR"/>
                <MSG_SEQ_FLG type="CHAR"/>
                <PROCESS_INSTANCE type="NUMBER"/>
                <PUBLISH_RULE_ID type="CHAR"/>
                <MSGNODENAME type="CHAR"/>
            </PSCAMA>
        </FieldTypes>
        <MsgData>
            <Transaction>
                <ADDRESS_SBR class="R">
                    <CITY><xsl:value-of select="CITY"/></CITY>
                    <STATE><xsl:value-of select="STATE"/></STATE>
                </ADDRESS_SBR>
                <PSCAMA class="R">
                    <LANGUAGE_CD>ENG</LANGUAGE_CD>
                    <AUDIT_ACTN>C</AUDIT_ACTN>
                    <BASE_LANGUAGE_CD>ENG</BASE_LANGUAGE_CD>
                    <MSG_SEQ_FLG/>
                    <PROCESS_INSTANCE>0</PROCESS_INSTANCE>
                    <PUBLISH_RULE_ID/>
                    <MSGNODENAME/>
                </PSCAMA>
            </Transaction>
        </MsgData>
    </ZIP_RESP_XSL>
</xsl:template>

</xsl:stylesheet>
```

Navigate to Node definitions and select the web service provider node.



Nodes

Enter any information you have and click Search. Leave fields blank for a list of all values.

Find an Existing Value | Add a New Value

Search by: Node Name begins with US

[ ] Include History [✓] Correct History

Search | Advanced Search

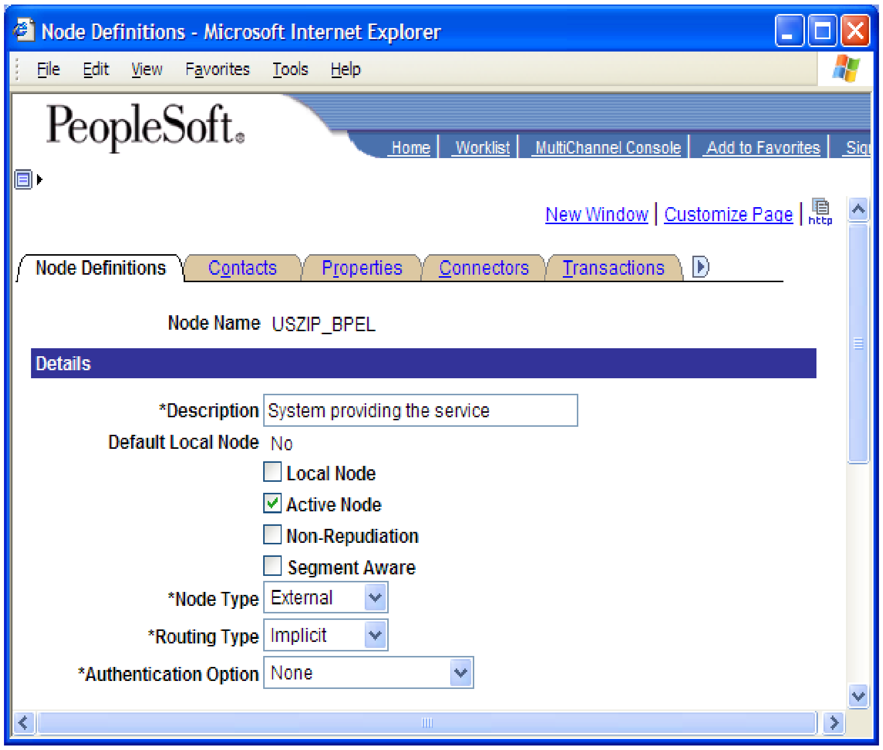
Search Results

View All First 1 of 1 Last

Node Name	Description	Local Node	Default Local Node	Node Type
USZIP_BPEL	System providing the service	0	N	PIA

Find an Existing Value | Add a New Value

If the WSDL Import Wizard created the BPEL / 3rd party node for you, change the Node Type to External.

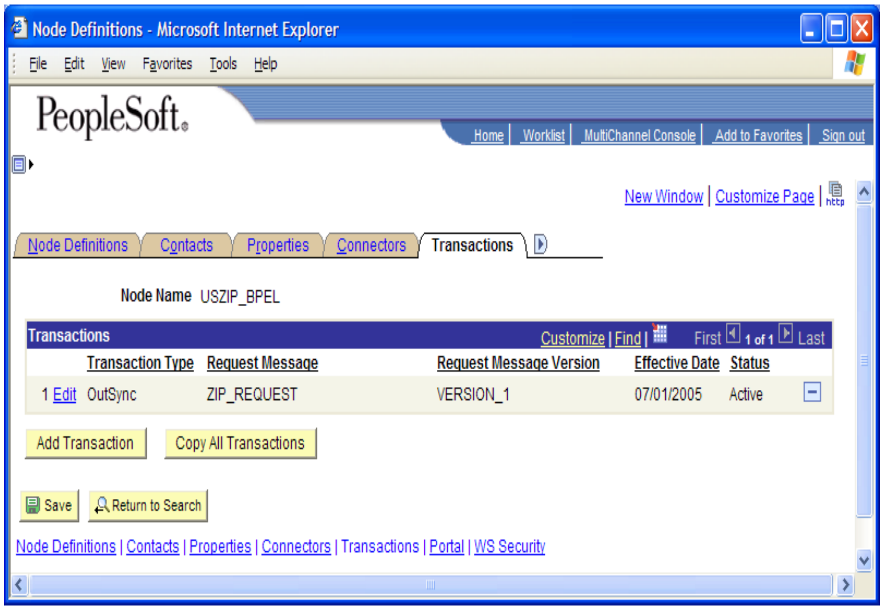

Node Definitions - Microsoft Internet Explorer
File Edit View Favorites Tools Help
PeopleSoft.
Home | Worklist | MultiChannel Console | Add to Favorites | Sig
New Window | Customize Page
Node Definitions
Contacts
Properties
Connectors
Transactions

Node Name USZIP_BPEL

Details

*Description	System providing the service
Default Local Node	No
	☐ Local Node
	✓ Active Node
	☐ Non-Repudiation
	☐ Segment Aware
*Node Type	External
*Routing Type	Implicit
*Authentication Option	None

Click the Transactions tab

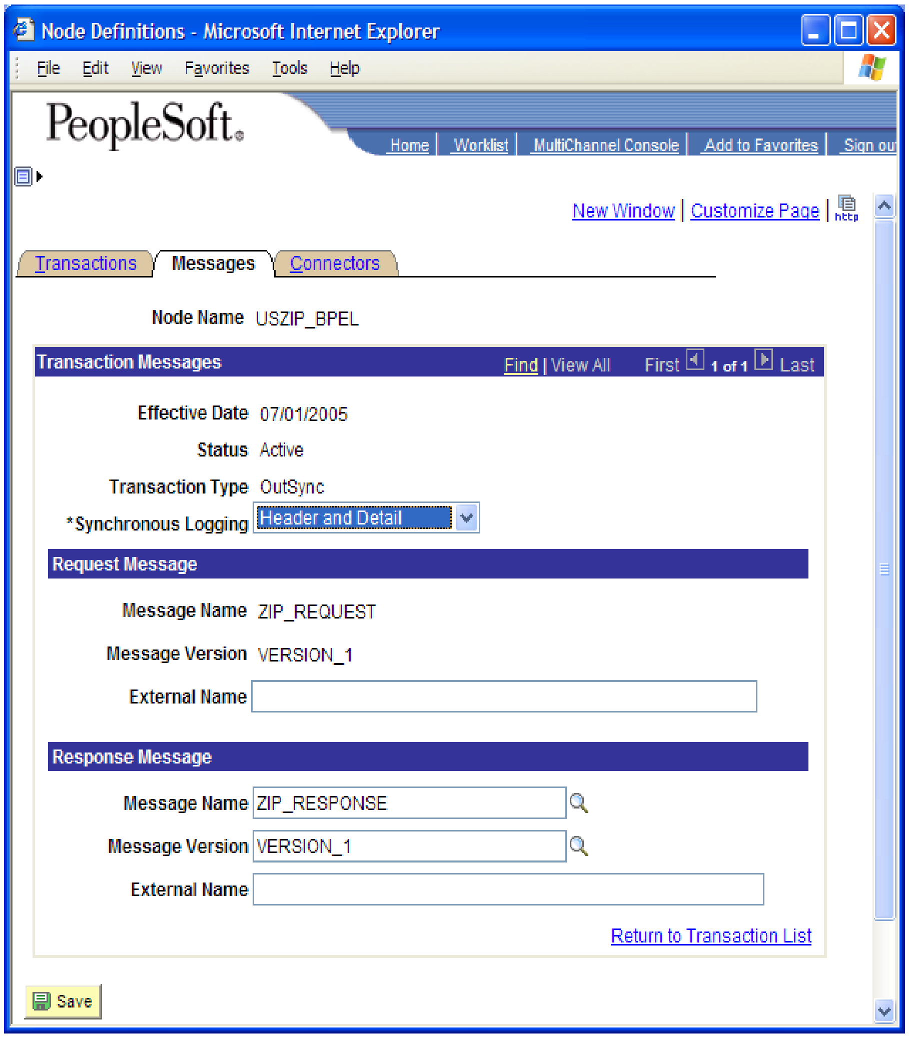

Node Definitions - Microsoft Internet Explorer
File Edit View Favorites Tools Help
PeopleSoft.
Home | Worklist | MultiChannel Console | Add to Favorites | Sign out
New Window | Customize Page
Node Definitions
Contacts
Properties
Connectors
Transactions
Customize | Find | First 1 of 1 Last
Add Transaction
Copy All Transactions
Save
Return to Search
Node Definitions | Contacts | Properties | Connectors | Transactions | Portal | WS Security

Node Name USZIP_BPEL

Transactions

	Transaction Type	Request Message	Request Message Version	Effective Date	Status
1 Edit	OutSync	ZIP_REQUEST	VERSION_1	07/01/2005	Active

Optional: Edit the transaction that the WSDL Import Wizard created for you to show full logging. This will allow you to see the XML being sent between PeopleSoft and the web service provider.


Node Definitions - Microsoft Internet Explorer
File Edit View Favorites Tools Help
PeopleSoft.
Home | Worklist | MultiChannel Console | Add to Favorites | Sign out
New Window | Customize Page
Transactions
Messages
Connectors
Find | View All
First 1 of 1 Last
Return to Transaction List
Save

Node Name USZIP_BPEL

Transaction Messages

Effective Date	07/01/2005
Status	Active
Transaction Type	OutSync
*Synchronous Logging	Header and Detail

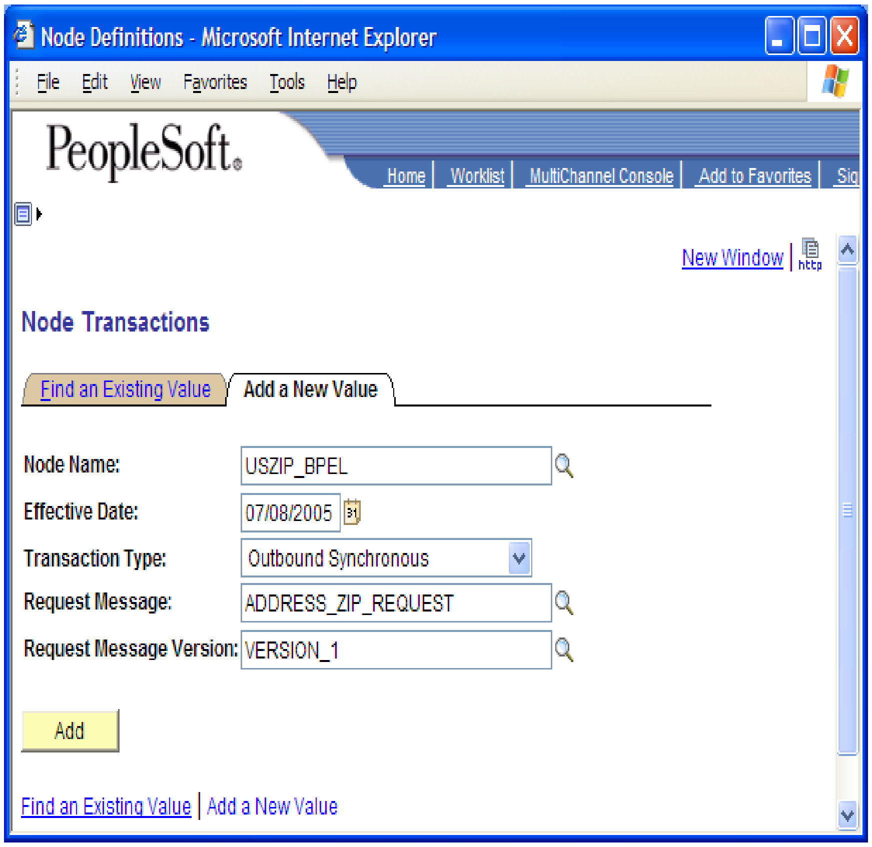
Request Message

Message Name	ZIP_REQUEST
Message Version	VERSION_1
External Name	

Response Message

Message Name	ZIP_RESPONSE
Message Version	VERSION_1
External Name	

On the Transactions tab, click Add Transaction to add the integration metadata for the structured messages. In this example, the transaction type is outbound synchronous. For consuming asynchronous web services in PeopleSoft, specify outbound asynchronous. Also notice that you are creating the transaction on the service provider node, not the PT_LOCAL or your local node.


Node Definitions - Microsoft Internet Explorer
File Edit View Favorites Tools Help
PeopleSoft.
Home | Worklist | MultiChannel Console | Add to Favorites | Sig
New Window |

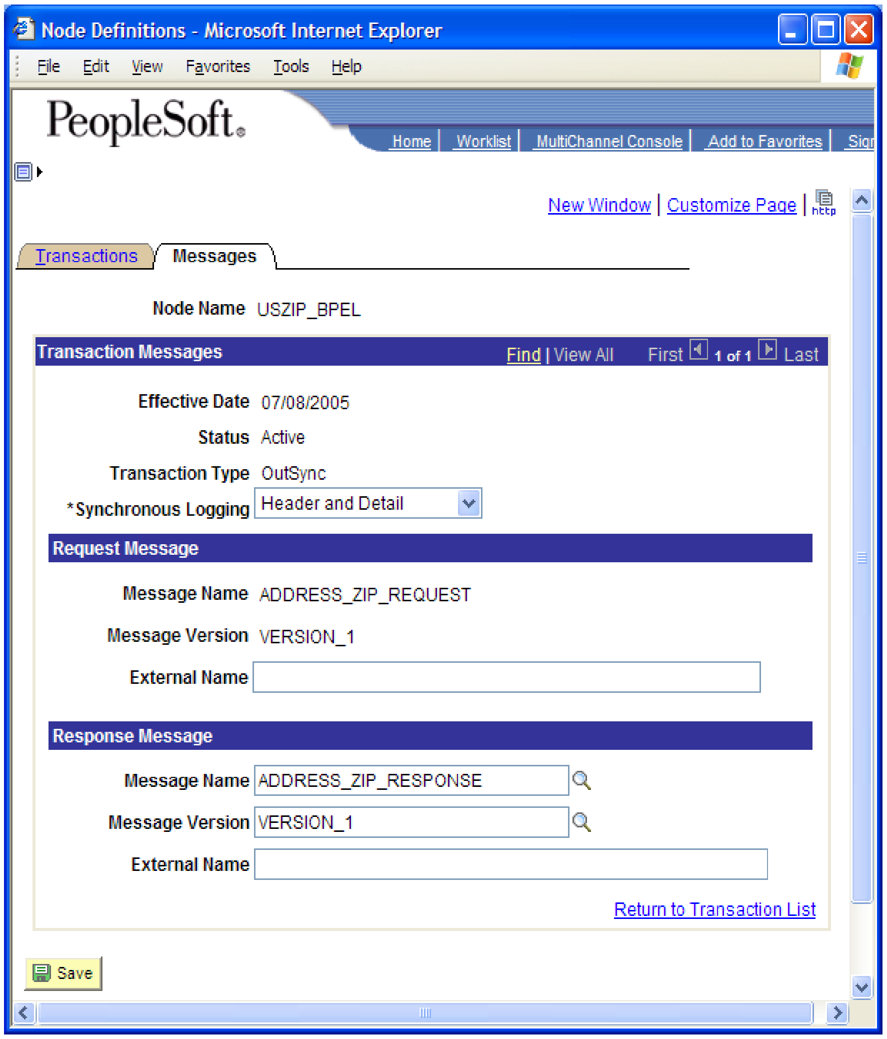
Node Transactions

Find an Existing Value | Add a New Value

Node Name:	USZIP_BPEL
Effective Date:	07/08/2005
Transaction Type:	Outbound Synchronous
Request Message:	ADDRESS_ZIP_REQUEST
Request Message Version:	VERSION_1

Add
Find an Existing Value | Add a New Value

Click Add and select the Messages tab. Set the Synchronous Logging to Header and Detail so you can see the XML that is exchanged between PeopleSoft and the web service provider. Add the response message and version and then click Save.


Node Definitions - Microsoft Internet Explorer
File Edit View Favorites Tools Help
PeopleSoft.
Home | Worklist | MultiChannel Console | Add to Favorites | Sign
New Window | Customize Page |
Transactions
Messages
Find | View All First 1 of 1 Last
Return to Transaction List
Save

Node Name USZIP_BPEL

Transaction Messages

Effective Date	07/08/2005
Status	Active
Transaction Type	OutSync
*Synchronous Logging	Header and Detail

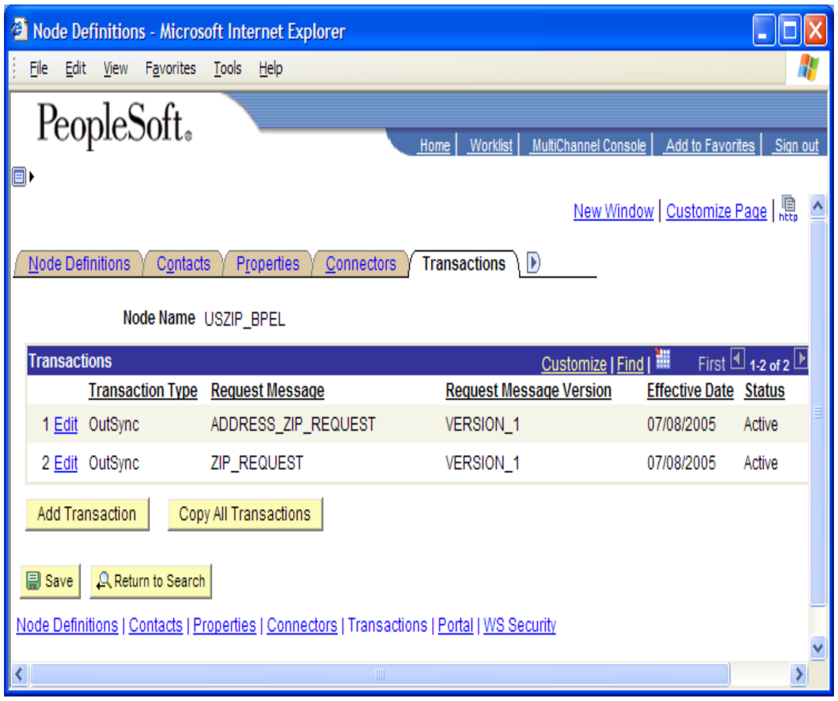
Request Message

Message Name	ADDRESS_ZIP_REQUEST
Message Version	VERSION_1
External Name	

Response Message

Message Name	ADDRESS_ZIP_RESPONSE
Message Version	VERSION_1
External Name	

When you click the Return to Transaction List you will see two transactions on the page.


Node Definitions - Microsoft Internet Explorer
File Edit View Favorites Tools Help
PeopleSoft.
Home | Worklist | MultiChannel Console | Add to Favorites | Sign out
New Window | Customize Page |
Node Definitions | Contacts | Properties | Connectors | Transactions

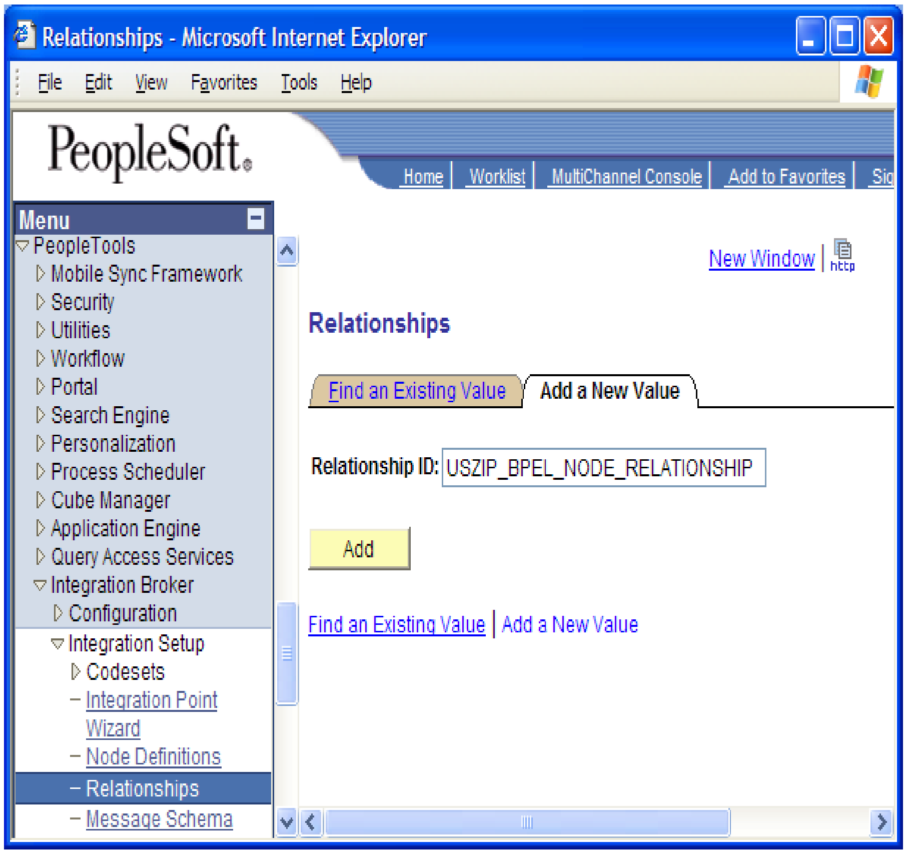
Node Name USZIP_BPEL

Transactions

	Transaction Type	Request Message	Request Message Version	Effective Date	Status
1 Edit	OutSync	ADDRESS_ZIP_REQUEST	VERSION_1	07/08/2005	Active
2 Edit	OutSync	ZIP_REQUEST	VERSION_1	07/08/2005	Active

Customize | Find | First 1-2 of 2
Add Transaction
Copy All Transactions
Save
Return to Search
Node Definitions | Contacts | Properties | Connectors | Transactions | Portal | WS Security

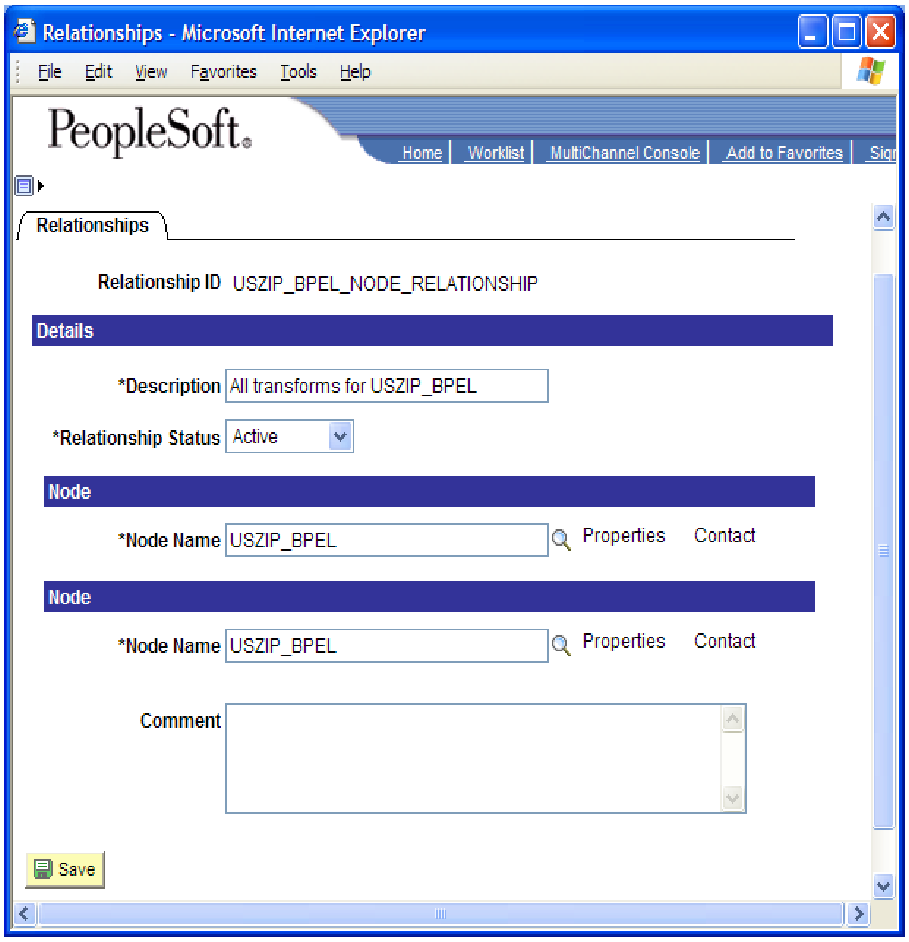
You now must add the transaction modifier to transform the rowset-based ADDRESS_ZIP_REQUEST message to the non-rowset based ZIP_REQUEST SOAP message and to transform the ZIP_RESPONSE SOAP message into the ADDRESS_CITY_STATE_RESPONSE rowset-based message. To do so, navigate to the Relationships page and create a new relationship for the BPEL PM / web service provider node and click the Add a New Value tab. The Relationship name is completely arbitrary. It's a good practice to have it contain the node name. Note that you will need only one relationship for all transformations on any single node.



Relationships

Relationship ID: USZIP_BPEL_NODE_RELATIONSHIP

Specify the web service provider for both Node fields on the page, then click Save. The source node and destination node are always the same node unless you are creating the relationship on an IB Hub system.


Relationships - Microsoft Internet Explorer
File Edit View Favorites Tools Help
PeopleSoft.
Home | Worklist | MultiChannel Console | Add to Favorites | Sign
Relationships

Relationship ID USZIP_BPEL_NODE_RELATIONSHIP

Details

*Description	All transforms for USZIP_BPEL
*Relationship Status	Active

Node

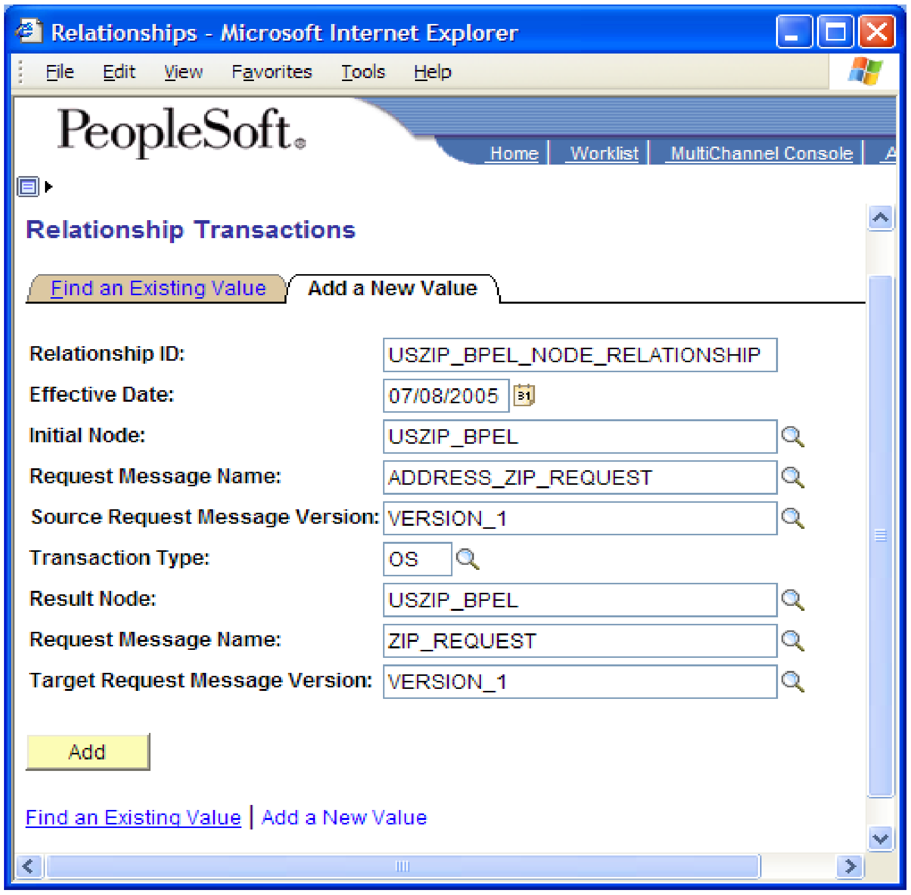
*Node Name USZIP_BPEL Properties Contact

Node

*Node Name USZIP_BPEL Properties Contact

Comment

Click Save and then click the Transaction Modifiers Tab.Click Add Transaction Modifier. Enter the values as shown below:


Relationships - Microsoft Internet Explorer
File Edit View Favorites Tools Help
PeopleSoft.
Home | Worklist | MultiChannel Console

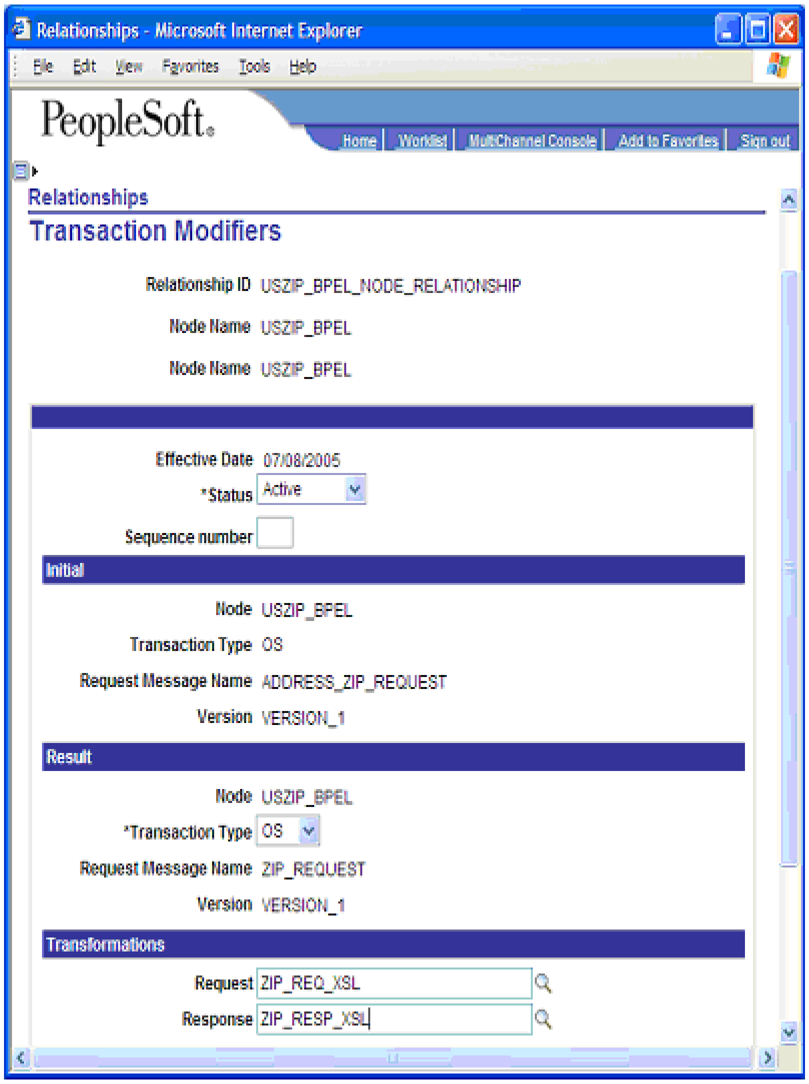
Relationship Transactions

Find an Existing Value | Add a New Value

Relationship ID:	USZIP_BPEL_NODE_RELATIONSHIP
Effective Date:	07/08/2005
Initial Node:	USZIP_BPEL
Request Message Name:	ADDRESS_ZIP_REQUEST
Source Request Message Version:	VERSION_1
Transaction Type:	OS
Result Node:	USZIP_BPEL
Request Message Name:	ZIP_REQUEST
Target Request Message Version:	VERSION_1

Both the initial and the resulting node are USZIP-BPEL. This is because the initial structured message request, ADDRESS_ZIP_REQUEST, is being sent to this node and so is the transformed result, ZIP_REQUEST. You can only choose OS, outbound synchronous, for the transaction type. Had this been a transaction modifier for an asynchronous web service, you would choose OA, outbound asynchronous.

Click Add and Set the Result Transaction Type to OS, outbound synchronous. It will be the only choice in the list. Then specify the names of the request and response Application Engine transformation programs, ZIP_REQ_XSL and ZIP_RESP_XSL. Click Save.


Relationships - Microsoft Internet Explorer
File Edit View Favorites Tools Help
PeopleSoft.
Home | Worklist | MultiChannel Console | Add to Favorites | Sign out

Relationships

Transaction Modifiers

Relationship ID	USZIP_BPEL_NODE_RELATIONSHIP
Node Name	USZIP_BPEL
Node Name	USZIP_BPEL
Effective Date	07/08/2005
*Status	Active
Sequence number	

Initial

Node	USZIP_BPEL
Transaction Type	OS
Request Message Name	ADDRESS_ZIP_REQUEST
Version	VERSION_1

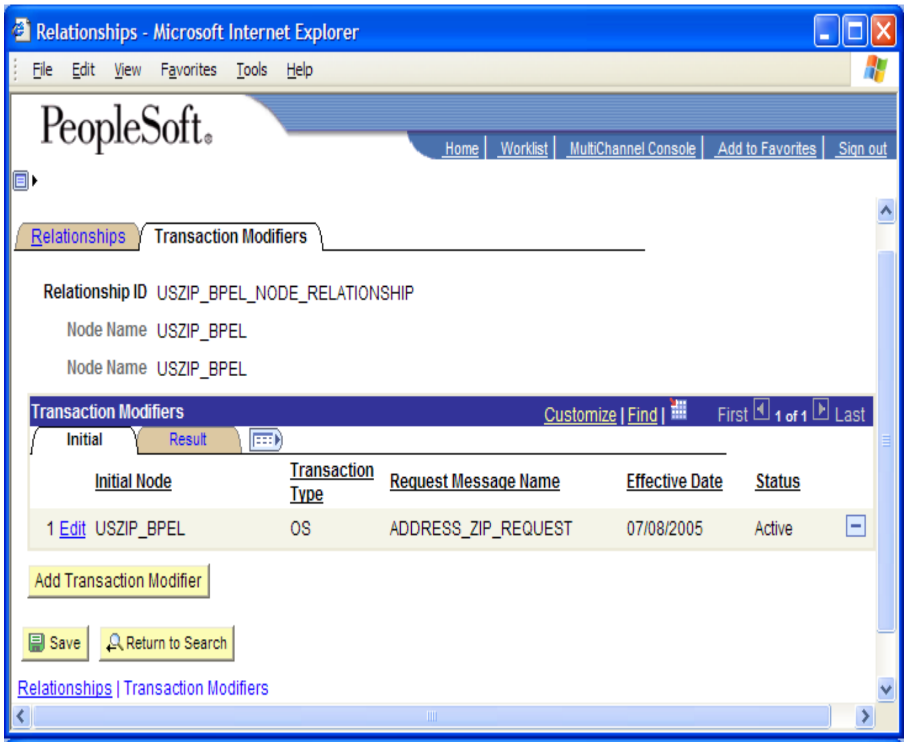
Result

Node	USZIP_BPEL
*Transaction Type	OS
Request Message Name	ZIP_REQUEST
Version	VERSION_1

Transformations

Request	ZIP_REQ_XSL
Response	ZIP_RESP_XSL

Click Save and then click the Return to Transaction List link. You will see the following Transaction Modifier:


Relationships - Microsoft Internet Explorer
File Edit View Favorites Tools Help
PeopleSoft.
Home | Worklist | MultiChannel Console | Add to Favorites | Sign out
Relationships
Transaction Modifiers

Relationship ID USZIP_BPEL_NODE_RELATIONSHIP

Node Name USZIP_BPEL

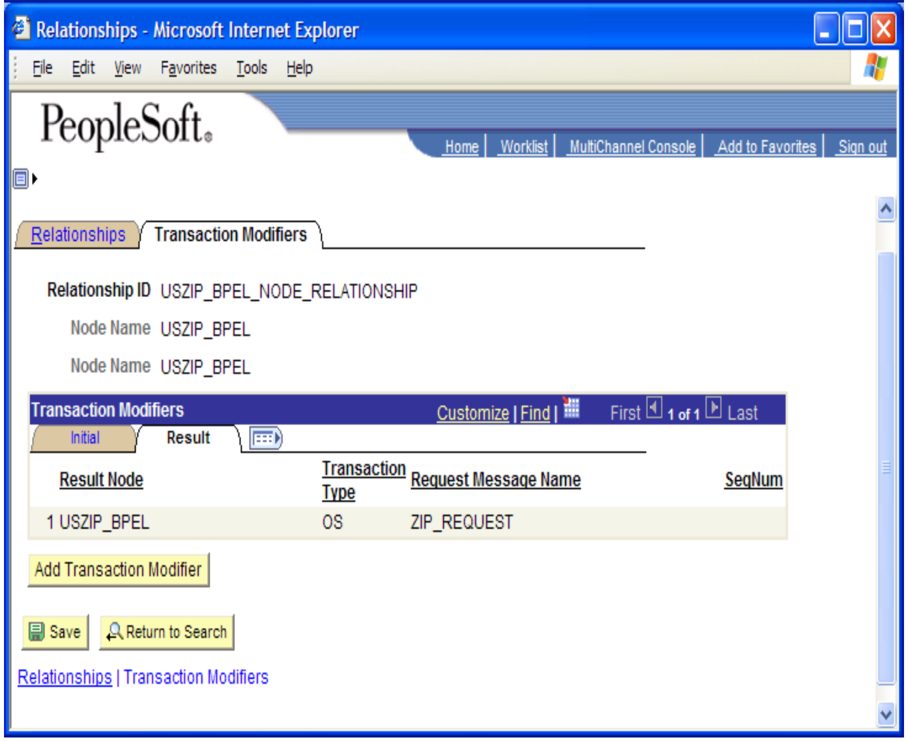
Node Name USZIP_BPEL

Transaction Modifiers — Customize | Find | First 1 of 1 Last

Initial | Result

	Initial Node	Transaction Type	Request Message Name	Effective Date	Status
1 Edit	USZIP_BPEL	OS	ADDRESS_ZIP_REQUEST	07/08/2005	Active

Add Transaction Modifier
Save
Return to Search
Relationships | Transaction Modifiers


Relationships - Microsoft Internet Explorer
File Edit View Favorites Tools Help
PeopleSoft.
Home | Worklist | MultiChannel Console | Add to Favorites | Sign out
Relationships
Transaction Modifiers

Relationship ID USZIP_BPEL_NODE_RELATIONSHIP

Node Name USZIP_BPEL

Node Name USZIP_BPEL

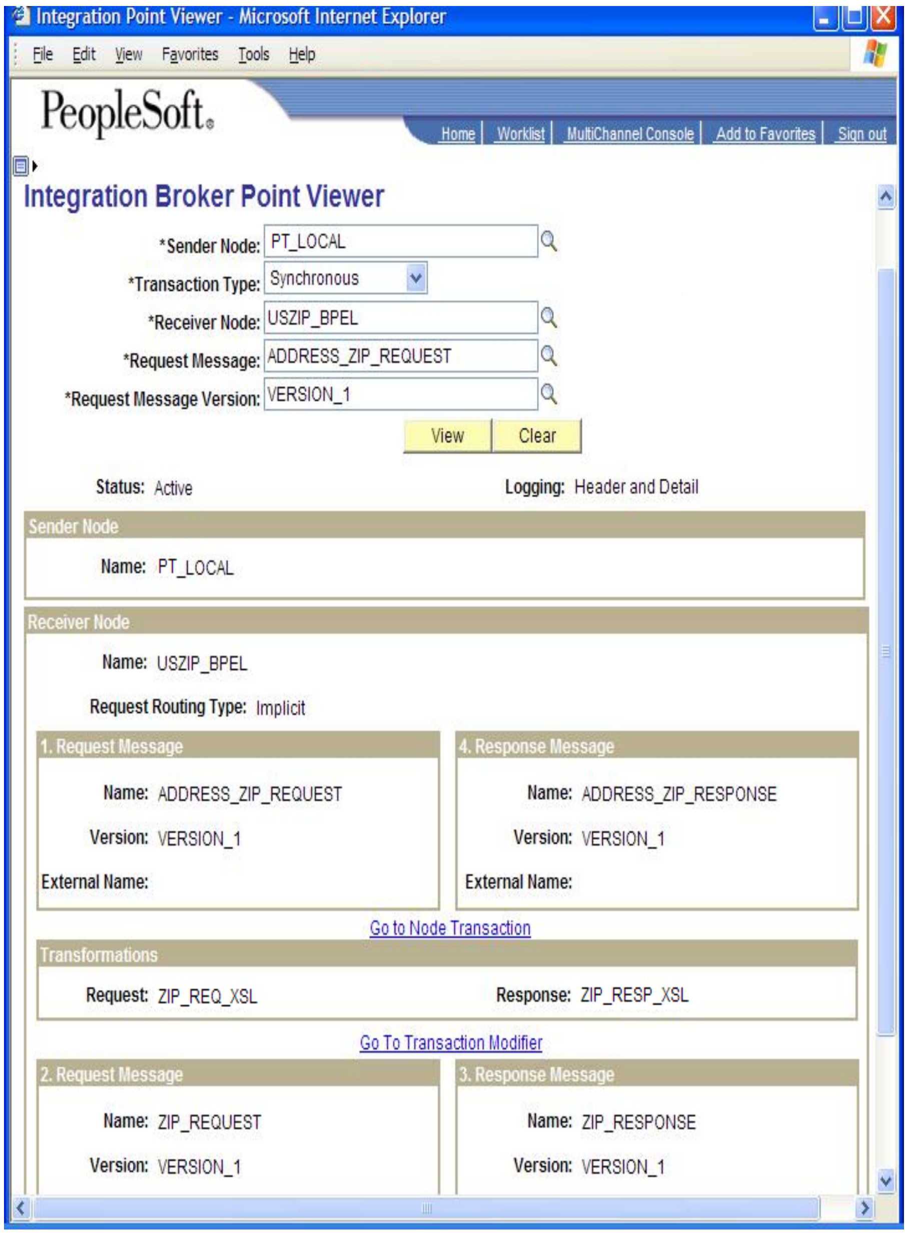
Transaction Modifiers — Customize | Find | First 1 of 1 Last

Initial | Result

	Result Node	Transaction Type	Request Message Name	SeqNum
1	USZIP_BPEL	OS	ZIP_REQUEST	

Add Transaction Modifier
Save
Return to Search
Relationships | Transaction Modifiers

Optional: you can verify that you created all the integration metadata correctly using the Integration Point Viewer:


Integration Point Viewer - Microsoft Internet Explorer
File Edit View Favorites Tools Help
PeopleSoft.
Home | Worklist | MultiChannel Console | Add to Favorites | Sign out
View
Clear
Go to Node Transaction
Go To Transaction Modifier

Integration Broker Point Viewer

*Sender Node: PT_LOCAL
*Transaction Type: Synchronous
*Receiver Node: USZIP_BPEL
*Request Message: ADDRESS_ZIP_REQUEST
*Request Message Version: VERSION_1

Status: Active
Logging: Header and Detail

Sender Node
Name: PT_LOCAL

Receiver Node
Name: USZIP_BPEL
Request Routing Type: Implicit

1. Request Message
Name: ADDRESS_ZIP_REQUEST
Version: VERSION_1
External Name:

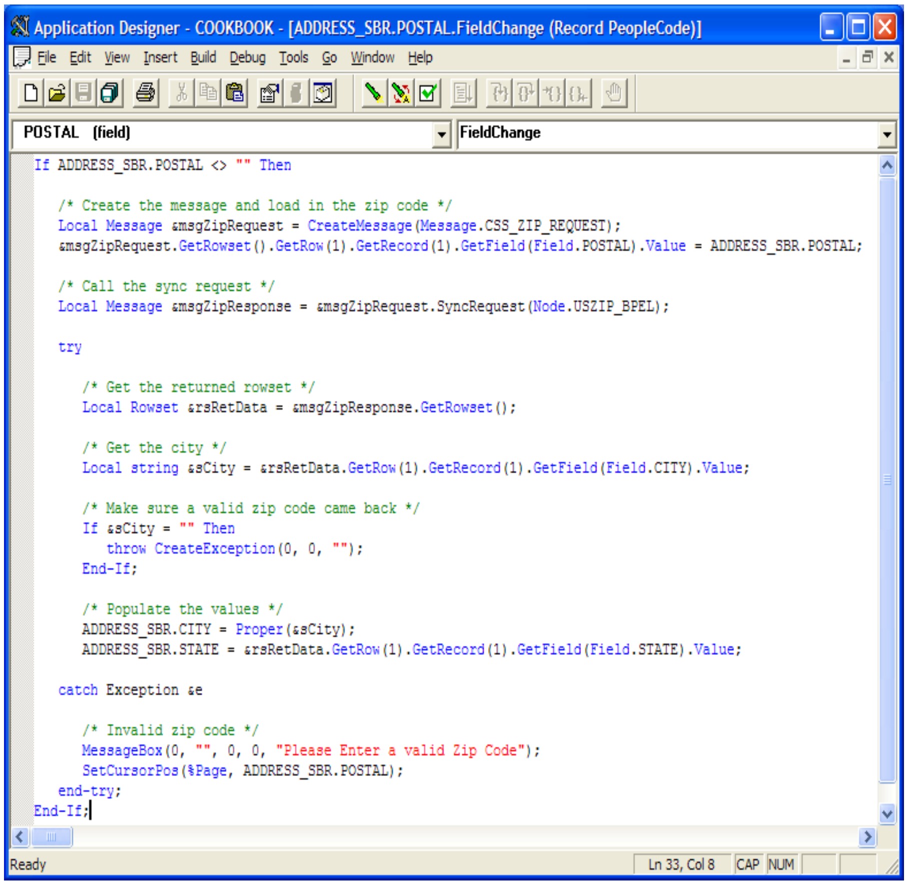
4. Response Message
Name: ADDRESS_ZIP_RESPONSE
Version: VERSION_1
External Name:

Transformations
Request: ZIP_REQ_XSL
Response: ZIP_RESP_XSL

2. Request Message
Name: ZIP_REQUEST
Version: VERSION_1

3. Response Message
Name: ZIP_RESPONSE
Version: VERSION_1

You are now ready to add the PeopleCode to your Component to invoke the web service. This particular web service will be invoked on a FieldChange event of the postal field. Initiating an asynchronous BPEL process is most likely to occur on the SavePostChange event of the Component or Record. The following is sample code for invoking the GetInfoByZip web service operation:


Application Designer - COOKBOOK - [ADDRESS_SBR.POSTAL.FieldChange (Record PeopleCode)]
File Edit View Insert Build Debug Tools Go Window Help
POSTAL (field)
FieldChange
Ready
Ln 33, Col 8
CAP
NUM

```
If ADDRESS_SBR.POSTAL <> "" Then

   /* Create the message and load in the zip code */
   Local Message &msgZipRequest = CreateMessage(Message.CSS_ZIP_REQUEST);
   &msgZipRequest.GetRowset().GetRow(1).GetRecord(1).GetField(Field.POSTAL).Value = ADDRESS_SBR.POSTAL;

   /* Call the sync request */
   Local Message &msgZipResponse = &msgZipRequest.SyncRequest(Node.USZIP_BPEL);

   try

      /* Get the returned rowset */
      Local Rowset &rsRetData = &msgZipResponse.GetRowset();

      /* Get the city */
      Local string &sCity = &rsRetData.GetRow(1).GetRecord(1).GetField(Field.CITY).Value;

      /* Make sure a valid zip code came back */
      If &sCity = "" Then
         throw CreateException(0, 0, "");
      End-If;

      /* Populate the values */
      ADDRESS_SBR.CITY = Proper(&sCity);
      ADDRESS_SBR.STATE = &rsRetData.GetRow(1).GetRecord(1).GetField(Field.STATE).Value;

   catch Exception &e

      /* Invalid zip code */
      MessageBox(0, "", 0, 0, "Please Enter a valid Zip Code");
      SetCursorPos(%Page, ADDRESS_SBR.POSTAL);
   end-try;
End-If;
```

```
If ADDRESS_SBR.POSTAL <> "" Then

  /* Create the message and load in the zip code */
  Local Message &msgZipRequest = CreateMessage(Message.CSS_ZIP_REQUEST);
  &msgZipRequest.GetRowset().GetRow(1).GetRecord(1).GetField(Field.POSTAL).Value =
ADDRESS_SBR.POSTAL;

  /* Call the sync request */
  Local Message &msgZipResponse = &msgZipRequest.SyncRequest(Node.USZIP_BPEL);

  try

    /* Get the returned rowset */
    Local Rowset &rsRetData = &msgZipResponse.GetRowset();

    /* Get the city */
    Local string &sCity = &rsRetData.GetRow(1).GetRecord(1).GetField(Field.CITY).Value;

    /* Make sure a valid zip code came back */
    If &sCity = "" Then
```

```
      throw CreateException(0, 0, "");
    End-If;

    /* Populate the values */
    ADDRESS_SBR.CITY = Proper(&sCity);
    ADDRESS_SBR.STATE = &rsRetData.GetRow(1).GetRecord(1).GetField(Field.STATE).Value;

  catch Exception &e

    /* Invalid zip code */
    MessageBox(0, "", 0, 0, "Please Enter a valid Zip Code");
    SetCursorPos(%Page, ADDRESS_SBR.POSTAL);
  end-try;
End-If;
```

You are now ready to consume an external web service or initiate a BPEL PM process from a PeopleSoft application.

For more information on importing WSDL and consuming BPEL and other external web services see the section, “Creating Third-Party Integrations Using WSDL” in Enterprise PeopleTools 8.46 PeopleBook: Integration Broker. You may also want to look at the section, “Developing Transformations.” Other sections in the same book worth reviewing are “Applying Transformations” and “Configuring Relationships.”

CHAPTER 5: HOW TO CONNECT BPEL AND JD EDWARDS APPLICATION

BPEL Connecting to WSG SOAP/RPC

Selecting desired Integration Point:

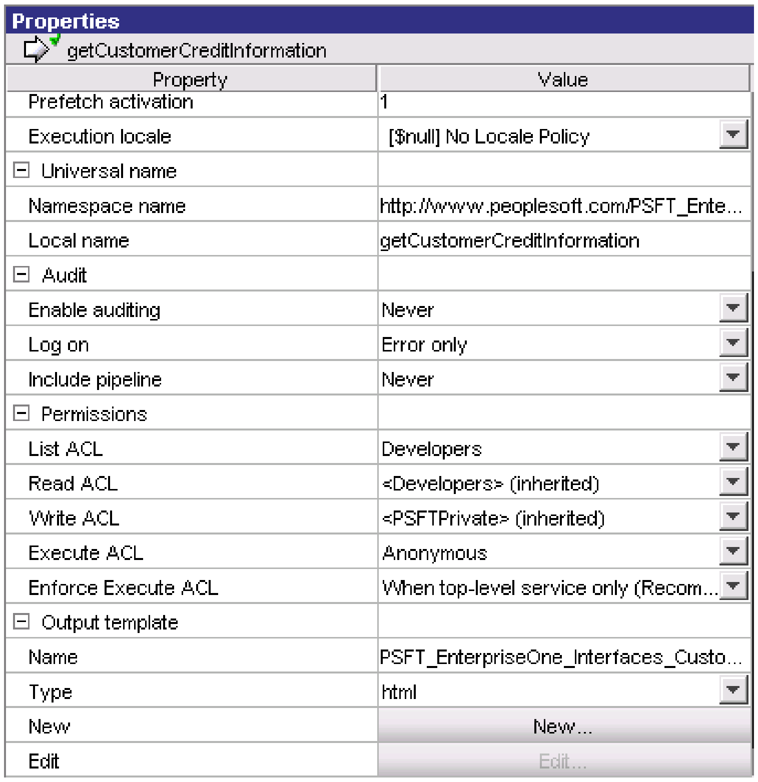
Assuming you have already installed the EnterpriseOne Integration Points packages, select the desired IP Flow that you want to expose as a web service. For the purpose of illustrating this process we have selected the Customer Credit Information Integration Point.

Verifying web service conformity for the Integration Point:

Using WSG Developer, verify all of the following properties by selecting the getCustomerCreditInformation flow in the PSFT_EntepriseOne_Interfaces package (PSFT_EnterpriseOne_Interfaces.PSFT_EnterpriseOne_Interfaces.Customer.getCustomerCreditInformation.getCustomerCreditInformation).

Verify that all the required properties for the Flow are set:

> **Property Group**: Universal Name
>
> **Property Name**: Namespace name – Verify that a globally unique namespace name like the name of company. This is critical for creating a web service. This name will need to be used at various places in the process.
>
> **Property Name**: Local Name – Verify that a meaningful name to identify service locally was provided.
>
> **Property**: Execute ACL
>
> Verify that the Execute ACL property is set to Anonymous



Properties

getCustomerCreditInformation

Property	Value
Prefetch activation	1
Execution locale	[$null] No Locale Policy
⊟ Universal name	
Namespace name	http://www.peoplesoft.com/PSFT_Ente...
Local name	getCustomerCreditInformation
⊟ Audit	
Enable auditing	Never
Log on	Error only
Include pipeline	Never
⊟ Permissions	
List ACL	Developers
Read ACL	<Developers> (inherited)
Write ACL	<PSFTPrivate> (inherited)
Execute ACL	Anonymous
Enforce Execute ACL	When top-level service only (Recom...
⊟ Output template	
Name	PSFT_EnterpriseOne_Interfaces_Custo...
Type	html
New	New...
Edit	Edit...

Note: Whenever the package is deployed to a new IS, all permissions are changed back to default (doesn't retain anonymous). So a customer will need to manually set this to anonymous each time the package is deployed on a new IS.

We will have to either pass authentication information from BPEL PM or create a new Java service that automatically sets the permissions each time the service is loaded.

Generating SOAP-RPC style WSDL:

To generate the WSDL for the service, choose Tools – Generate WSDL

Set the values in the Generate dialog box as follows:

- Host name to machine name or IP. Do not use localhost.
- Set protocol to SOAP-RPC
- Directive should remain default.
- Set the value of Target namespace to the XML namespace used for the service.
- Click OK. Choose the destination folder to save the file. WSG Developer will display a message that WSDL was successfully saved.

Afer generating the WSDL file, extract the schema definition of the WSDL to a separate .xsd file, for better usability. Name the XSD CustomerCreditInformation.xsd. Then, reference the xsd file in the WSDL as follows

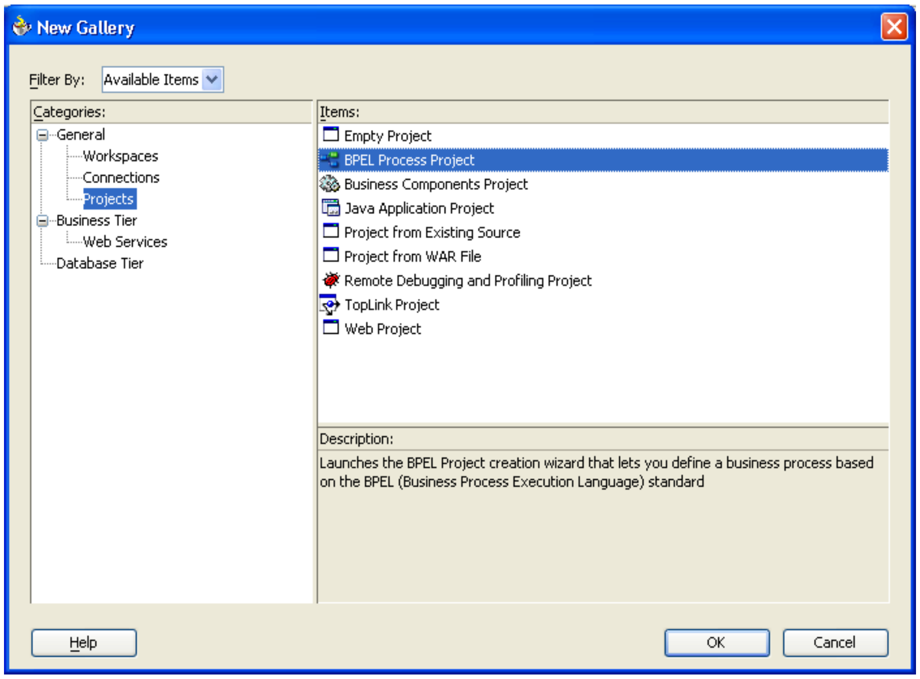
```
“ <?xml version="1.0" encoding="UTF-8"?>
        <wsdl:definitions
name="PSFT_EnterpriseOne_Interfaces_Customer_getCustomerCreditInformation"
targetNamespace="http://den-sa246967B/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
        xmlns:wsdns1="http://localhost/PSFT_EnterpriseOne_Interfaces/Customer/getCustomerCredit
Information/getCustomerCreditInformation"
        xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
        xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
        xmlns:tns="http://den-sa246967B/">
        <wsdl:types>
                <xsd:import schemaLocation="  CustomerCreditInformation.xsd"/>
        </wsdl:types>
```

You are now ready to consume the EnterpriseOne Integration Point provided web service.

Generating BPEL PM process to consume the service

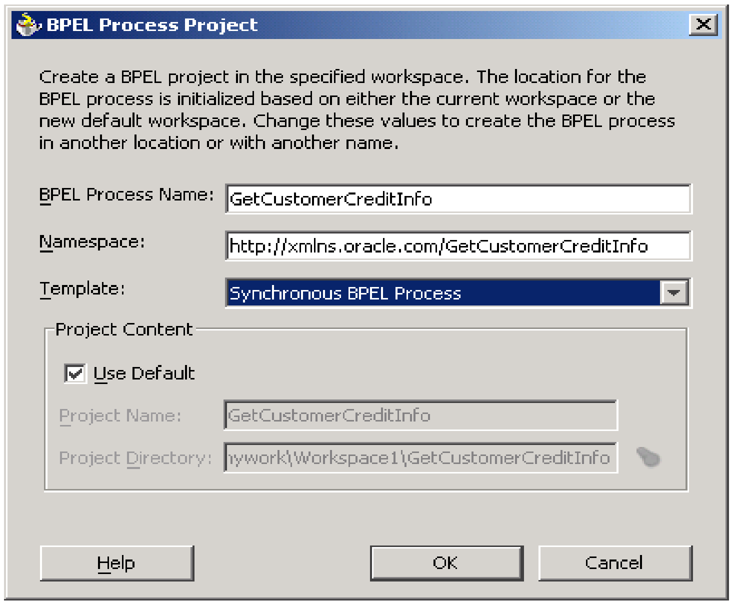
The following section assumes that BPEL PM server and designer are properly set up and that the appropriate workspace and connections are created in the designer.

Start BPEL PM server. Launch BPEL Designer. Right-click on the workspace and choose New Project. From Project Items choose BPEL Process Project and click OK.



Description:
Launches the BPEL Project creation wizard that lets you define a business process based on the BPEL (Business Process Execution Language) standard

In the project properties dialog, select process name, and choose Synchronous service from the Template dropdown. Click OK.



Create a BPEL project in the specified workspace. The location for the BPEL process is initialized based on either the current workspace or the new default workspace. Change these values to create the BPEL process in another location or with another name.

BPEL Process Name: GetCustomerCreditInfo

Namespace: http://xmlns.oracle.com/GetCustomerCreditInfo

Template: Synchronous BPEL Process

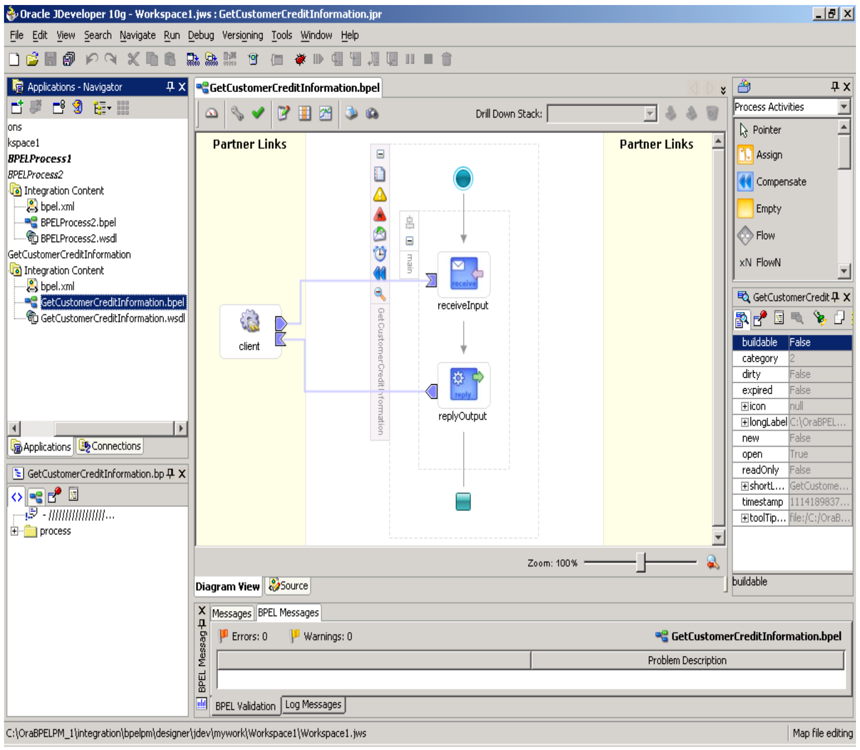
Project Content

☑ Use Default

Project Name: GetCustomerCreditInfo

Project Directory: ywork\Workspace1\GetCustomerCreditInfo

The following sections appear. If this does not appear, click Diagram View and double click GetCustomerCreditInfo in the Applications Navigator section. See the *Oracle BPEL Process Manager Developer's Guide* for a description of these sections.



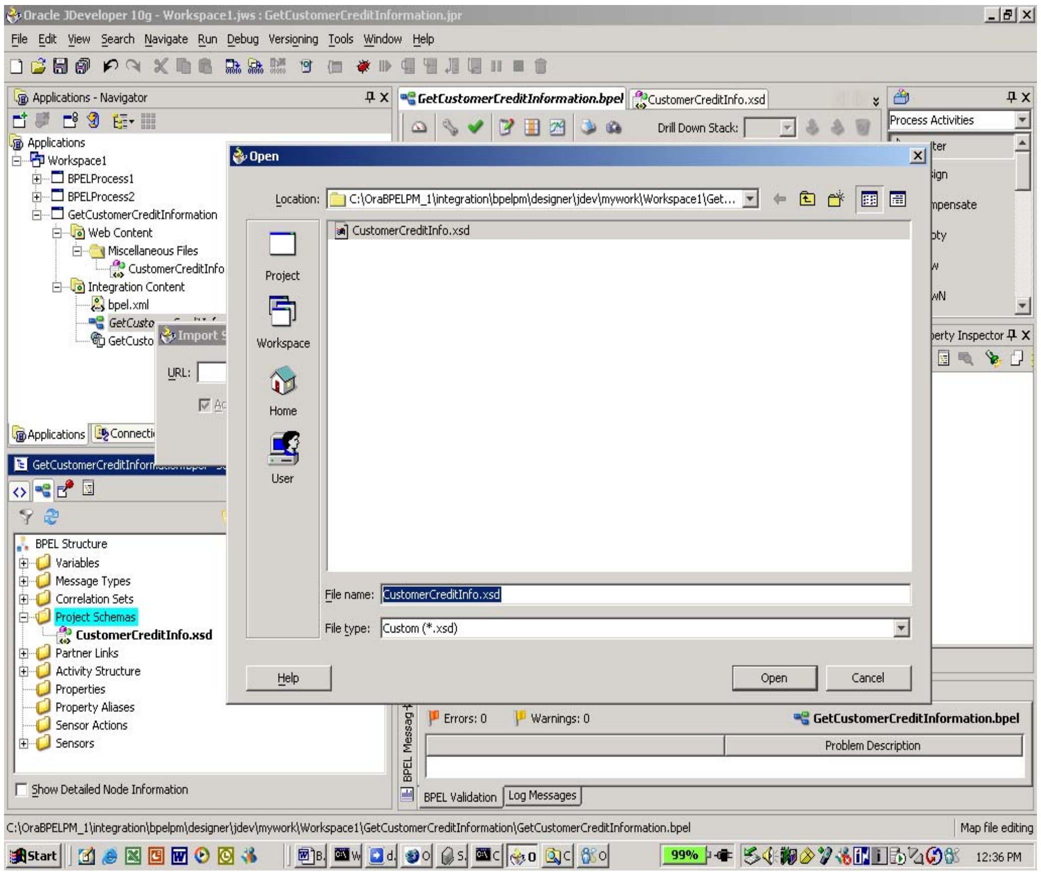
buildable	False
category	2
dirty	False
expired	False
⊞icon	null
⊞longLabel	C:\OraBPEL...
new	False
open	True
readOnly	False
⊞shortL...	GetCustome...
timestamp	1114189837...
⊞toolTip...	file:/C:/OraB...

Errors: 0 Warnings: 0 GetCustomerCreditInformation.bpel

C:\OraBPELPM_1\integration\bpelpm\designer\jdev\mywork\Workspace1\Workspace1.jws

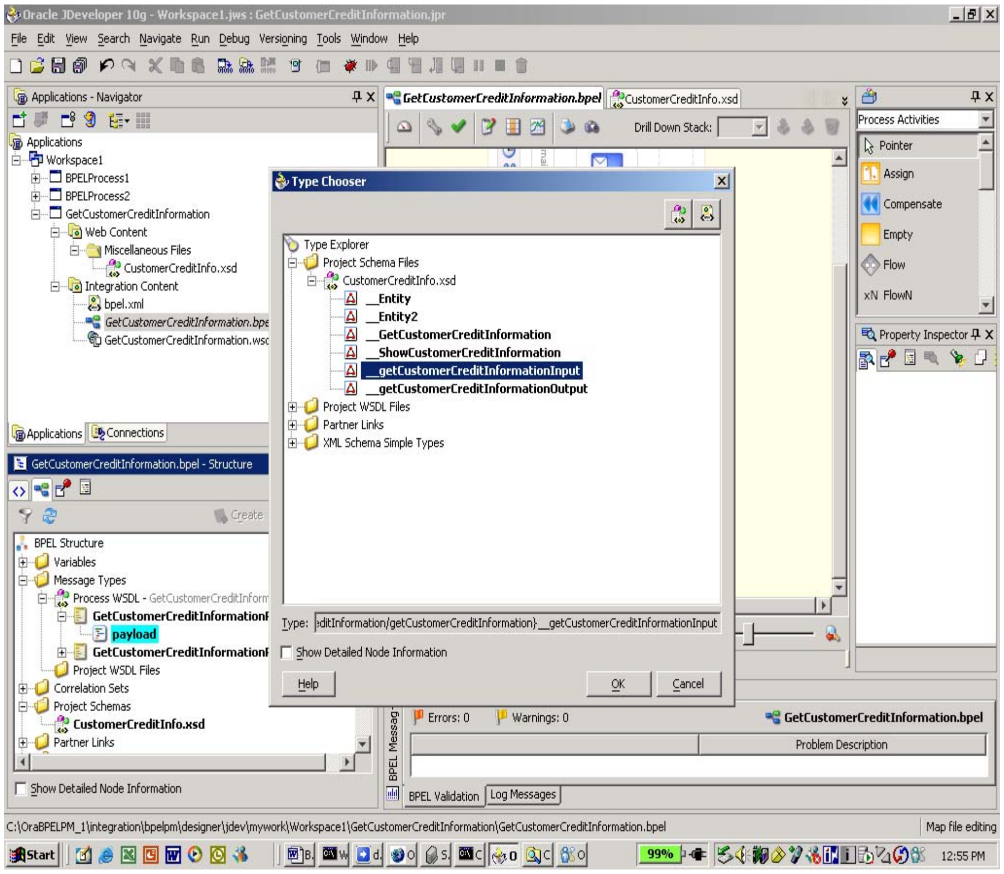
Browse to the project directory from explorer and copy both the WSDL and XSD files that you created using WSG Developer to this project folder.

Import the GetCustomerCreditInfo Schema into the project. In the Structure window, right-click on Project Schemas and select Import Schemas. In the Import Schema window, browse the file system and select the CustomerCreditInformation.xsd. Select ok, ok, and save all.

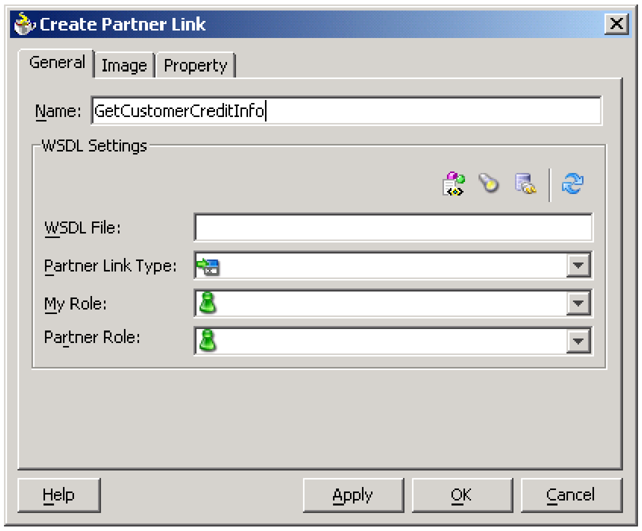

Open
Location: C:\OraBPELPM_1\integration\bpelpm\designer\jdev\mywork\Workspace1\Get...
CustomerCreditInfo.xsd
Project
Workspace
Home
User
File name: CustomerCreditInfo.xsd
File type: Custom (*.xsd)
Help
Open
Cancel

Associate Schema to Request Message type. In the Structure window, select Message Types > Process WSDL > GetCustomerCreditInformationRequestMessage > payload.

Right-click on payload and select Edit Message Part. Select the type radio button, and click the browse types button. In the Type Chooser, select Project Schema Files > CustomerCreditInformation.xsd > __getCustomerCreditInformationInput. Click ok, ok, and save all.


Oracle JDeveloper 10g - Workspace1.jws : GetCustomerCreditInformation.jpr
Type Chooser
Type Explorer
Project Schema Files
CustomerCreditInfo.xsd
__Entity
__Entity2
__GetCustomerCreditInformation
__ShowCustomerCreditInformation
__getCustomerCreditInformationInput
__getCustomerCreditInformationOutput
Project WSDL Files
Partner Links
XML Schema Simple Types
Type: :ditInformation/getCustomerCreditInformation}__getCustomerCreditInformationInput
Show Detailed Node Information
Help
OK
Cancel

Similarly, associate __getCustomerCreditInformationOutput with the Response Message to the Response Message type payload. Open the BPEL Editor by double clicking in the GetCustomerCreditInformation.bpel node in the Application Navigator. Drag and drop a **partner link** task in the BPEL process (in yellow area).



Name: GetCustomerCreditInfo

WSDL Settings

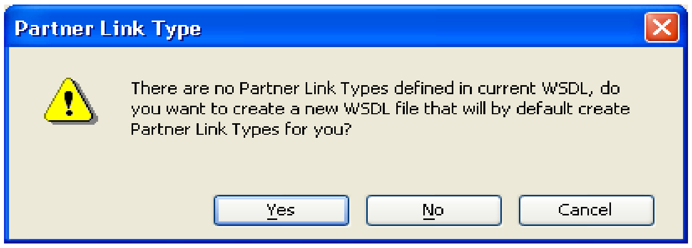
WSDL File:

Partner Link Type:

My Role:

Partner Role:

Provide partner link name. Click on the first icon in the WSDL Settings frame to select the WSDL file from the file system. Select the WSG WSDL copied to the project directory. Designer will display the following message:



There are no Partner Link Types defined in current WSDL, do you want to create a new WSDL file that will by default create Partner Link Types for you?

Select Yes to automatically create partner link types.

Designer will create a partner link type and populate it. Keep My Roles filed as Not specified and populate Partner Role with the value available in the drop down. Click OK to close the partner link setting and save the BPEL file.

Next follow the BPEL developer guide to add the following items to the BPEL file.

- An Assign activity to load the service variables
- An Invoke activity to invoke the GetCustomerCreditInfo IP service
- An Assign activity to assign the output of the service call to the Response message

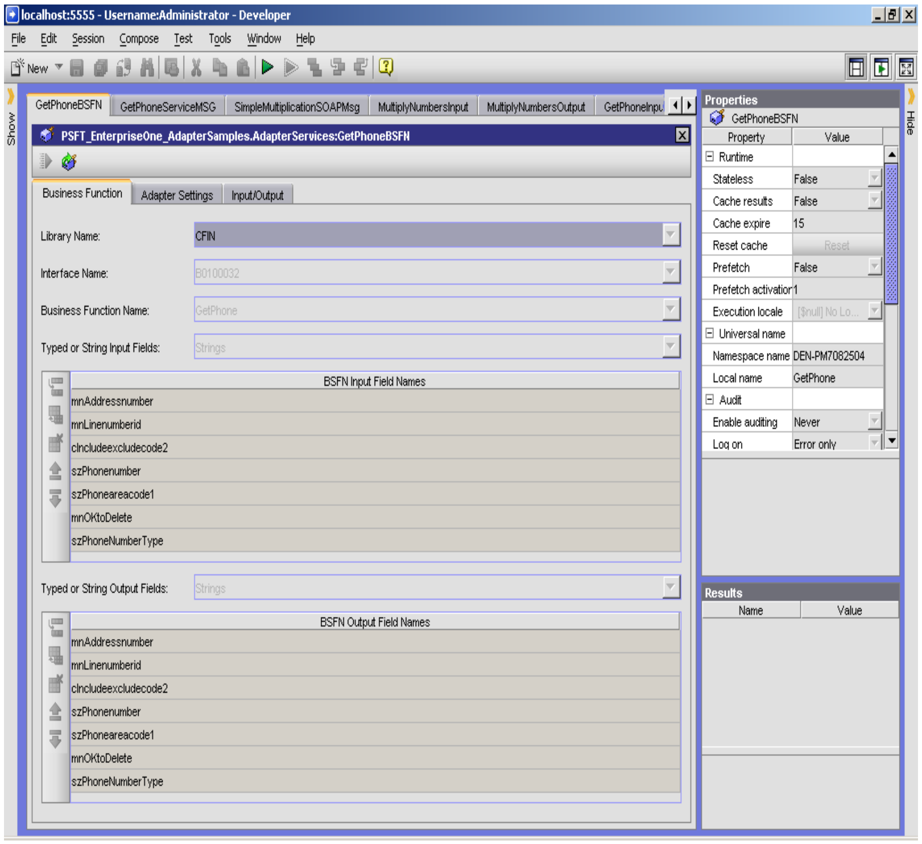
Follow the guide to validate the BPEL and save the file. To compile and deploy the BPEL process, right-click on project in Application Navigator. Select Deploy > *connection name* > Deploy to Default Domain. Click OK. This will compile and deploy the BPEL process. Watch the bottom of the screen for any errors. Refer to the *BPEL Developer's Guide* for more information.

BPEL Connecting to WSG SOAP/MSG

This section describes the process to generate the SOAP/MSG compatible service in WSG Developer and generating the WSDL for the same. It also describes the generation of BPEL PM Process for consuming this service. This section uses BSFN as an example adapter service. However, the same process can be followed for JDBJ as well as XAPI response service.

Generating SOAP/MSG compatible Flow service

Generate BSFN adapter service to call EnterpriseOne BSFN. Create this in the same way you would develop any EnterpriseOne BSFN adapter service. For example, consider the GetPhone Adapter Service which looks as follows:


localhost:5555 - Username:Administrator - Developer
File Edit Session Compose Test Tools Window Help
New
Show
Hide
GetPhoneBSFN GetPhoneServiceMSG SimpleMultiplicationSOAPMsg MultiplyNumbersInput MultiplyNumbersOutput GetPhoneInpu
Business Function Adapter Settings Input/Output
Properties
Results

PSFT_EnterpriseOne_AdapterSamples.AdapterServices:GetPhoneBSFN

Library Name: CFIN

Interface Name: B0100032

Business Function Name: GetPhone

Typed or String Input Fields: Strings

BSFN Input Field Names
mnAddressnumber
mnLinenumberid
cIncludeexcludecode2
szPhonenumber
szPhoneareacode1
mnOKtoDelete
szPhoneNumberType

Typed or String Output Fields: Strings

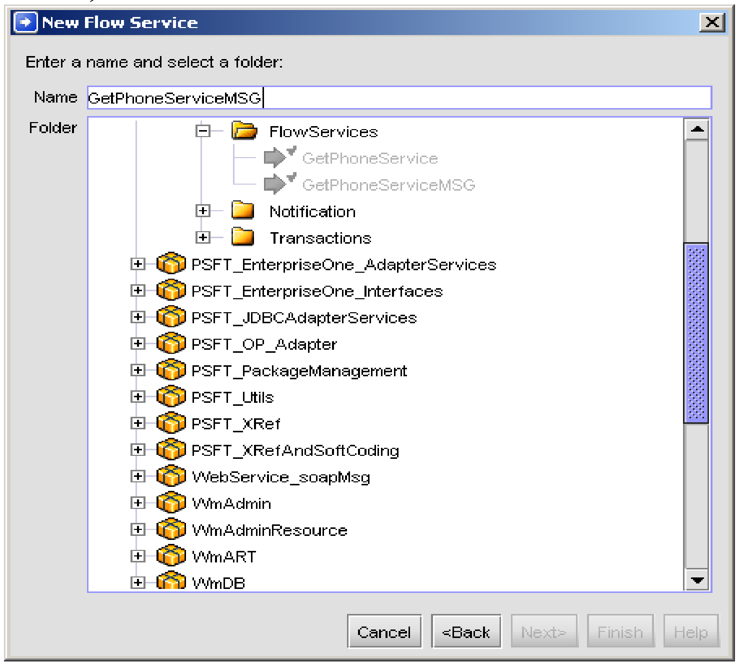
BSFN Output Field Names
mnAddressnumber
mnLinenumberid
cIncludeexcludecode2
szPhonenumber
szPhoneareacode1
mnOKtoDelete
szPhoneNumberType

GetPhoneBSFN

Property	Value
⊟ Runtime	
Stateless	False
Cache results	False
Cache expire	15
Reset cache	Reset
Prefetch	False
Prefetch activation	1
Execution locale	[$null] No Lo...
⊟ Universal name	
Namespace name	DEN-PM7082504
Local name	GetPhone
⊟ Audit	
Enable auditing	Never
Log on	Error only

Name	Value

Next, create a flow service. From File > New, choose Flow Service option. Give a meaningful name to the service, select destination folder and choose finish to create the flow service.



Enter a name and select a folder:

Name GetPhoneServiceMSG

Folder
- FlowServices
  - GetPhoneService
  - GetPhoneServiceMSG
- Notification
- Transactions
- PSFT_EnterpriseOne_AdapterServices
- PSFT_EnterpriseOne_Interfaces
- PSFT_JDBCAdapterServices
- PSFT_OP_Adapter
- PSFT_PackageManagement
- PSFT_Utils
- PSFT_XRef
- PSFT_XRefAndSoftCoding
- WebService_soapMsg
- WmAdmin
- WmAdminResource
- WmART
- WmDB

Double-click on the service just created in the navigation pane to open the editor for the service.

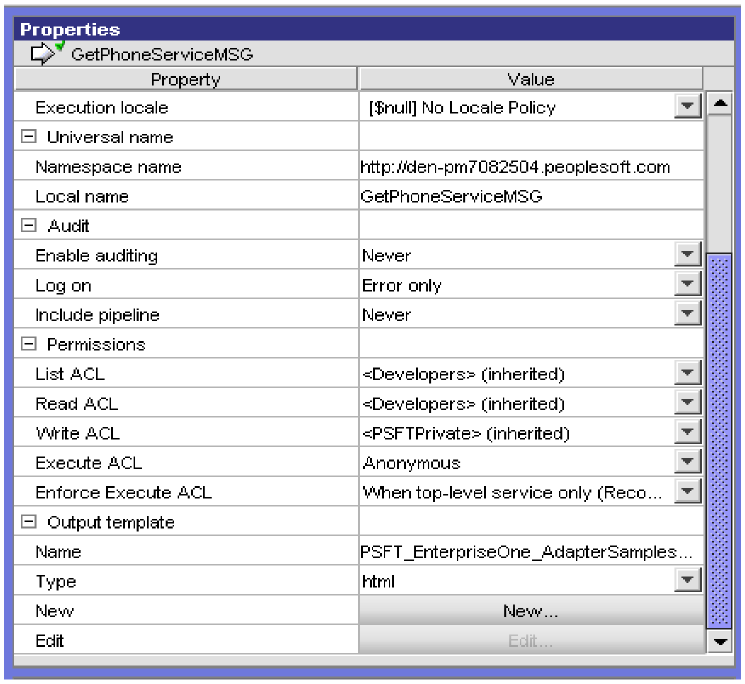
Set the following service properties:

Property Group: Universal Name

Property Name: Namespace name – Give a globally unique namespace name like the company name http://www.peoplesoft.com. This is critical for creating a web service. This name be used at various places in the process.

Property Name: Local Name – Provide a meaningful name to identify the service locally.

Property: Execute ACL – Anonymous

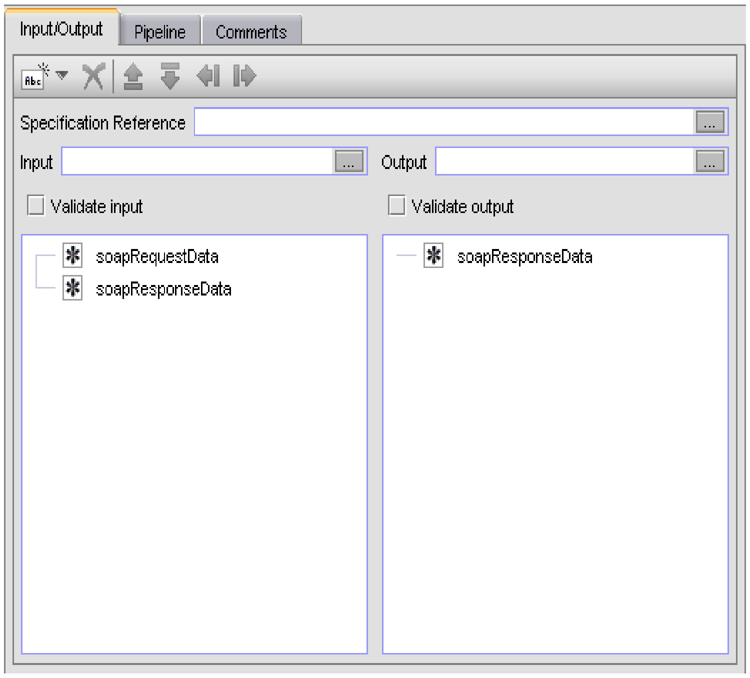


Properties

GetPhoneServiceMSG

Property	Value
Execution locale	[$null] No Locale Policy
⊟ Universal name	
Namespace name	http://den-pm7082504.peoplesoft.com
Local name	GetPhoneServiceMSG
⊟ Audit	
Enable auditing	Never
Log on	Error only
Include pipeline	Never
⊟ Permissions	
List ACL	<Developers> (inherited)
Read ACL	<Developers> (inherited)
Write ACL	<PSFTPrivate> (inherited)
Execute ACL	Anonymous
Enforce Execute ACL	When top-level service only (Reco...
⊟ Output template	
Name	PSFT_EnterpriseOne_AdapterSamples...
Type	html
New	New...
Edit	Edit...

Click on Input/Output tab in the service editor. From the list of available types, choose **object** to create two input objects: soapRequestData and soapResponseData. Create and output object soapResponseData.



Specification Reference

Input

Output

Validate input

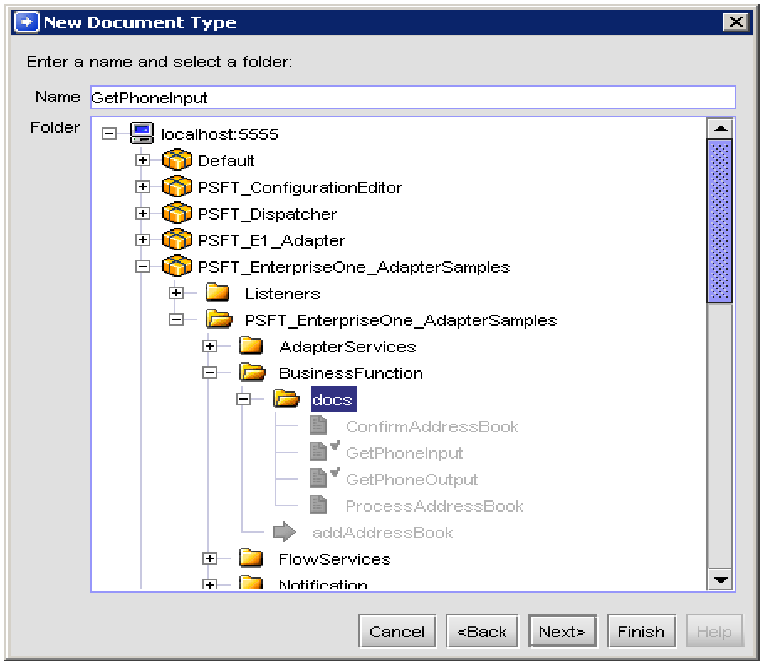
Validate output

soapRequestData

soapResponseData

soapResponseData

Next, create the input and output document types. Click on File > New to invoke the New Object wizard. Choose Document Type and click Next. Choose the document type name and location. Click Finish to create the document type.



Enter a name and select a folder:

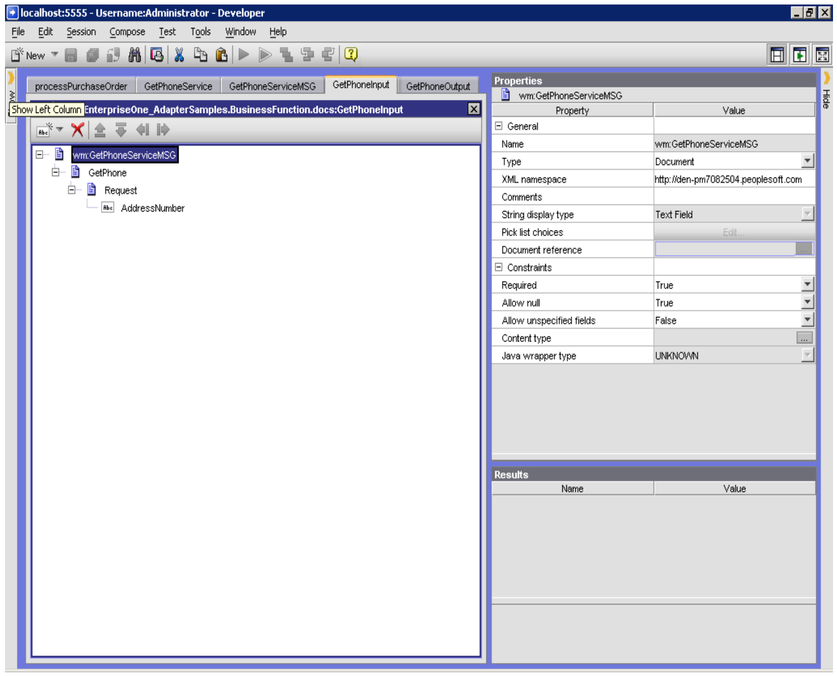
Name GetPhoneInput

Folder

```
localhost:5555
  Default
  PSFT_ConfigurationEditor
  PSFT_Dispatcher
  PSFT_E1_Adapter
  PSFT_EnterpriseOne_AdapterSamples
    Listeners
    PSFT_EnterpriseOne_AdapterSamples
      AdapterServices
      BusinessFunction
        docs
          ConfirmAddressBook
          GetPhoneInput
          GetPhoneOutput
          ProcessAddressBook
        addAddressBook
      FlowServices
      Notification
```

Double-click on document type in the navigation pane to open the editor. Create elements as follows:

- From the variable list choose Document element. This top level element must be named in a specific format. Choose any namespace name. This element has to be named as namespace name: flow service local name e.g. wm:GetPhoneServiceMSG. Click on the element and change its XML namespace property. This must have the same value as namespace name of service.
- Create another Document element as child of this root element. Name this element as document identifier. E.g. GetPhone.
- Create Document element Request under second document element.
- Create required input variables under request.

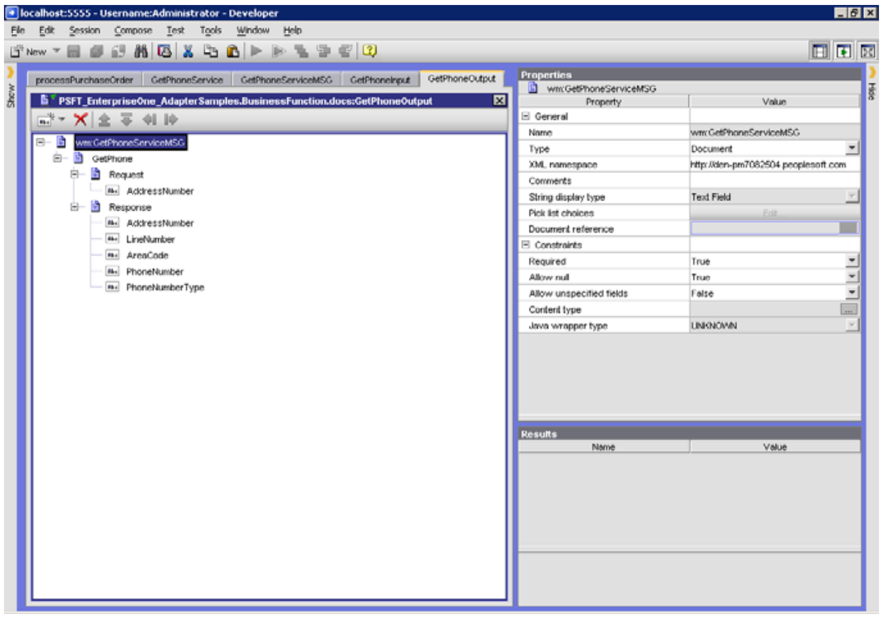


Property	Value
⊟ General	
Name	wm:GetPhoneServiceMSG
Type	Document
XML namespace	http://den-pm7082504.peoplesoft.com
Comments	
String display type	Text Field
Pick list choices	Edit...
Document reference	
⊟ Constraints	
Required	True
Allow null	True
Allow unspecified fields	False
Content type	
Java wrapper type	UNKNOWN

To create output document type, select File > New to invoke the New Object wizard. Choose Document Type and click Next. Choose the document type name and location. Click Finish to create the document type. Double-click on output document type in the navigation pane to open the editor. Create elements as follows:

- From the variable list choose Document element. This top level element has to be named in a specific format. Choose any namespace name. This element must be named as namespace name: flow service local name e.g wm:GetPhoneServiceMSG. Click on the element and change its XML namespace property. This must have the same value as namespace name of service.
- Create another Document element as child of this root element. Name this element as document identifier. E.g. GetPhone.
- Create Document element Request under second document element.
- Create same input variables under request as in 8.d.
- Create Dcoument element Response under second document element.

- Create required output variables under this document element.



localhost:5555 - Username:Administrator - Developer

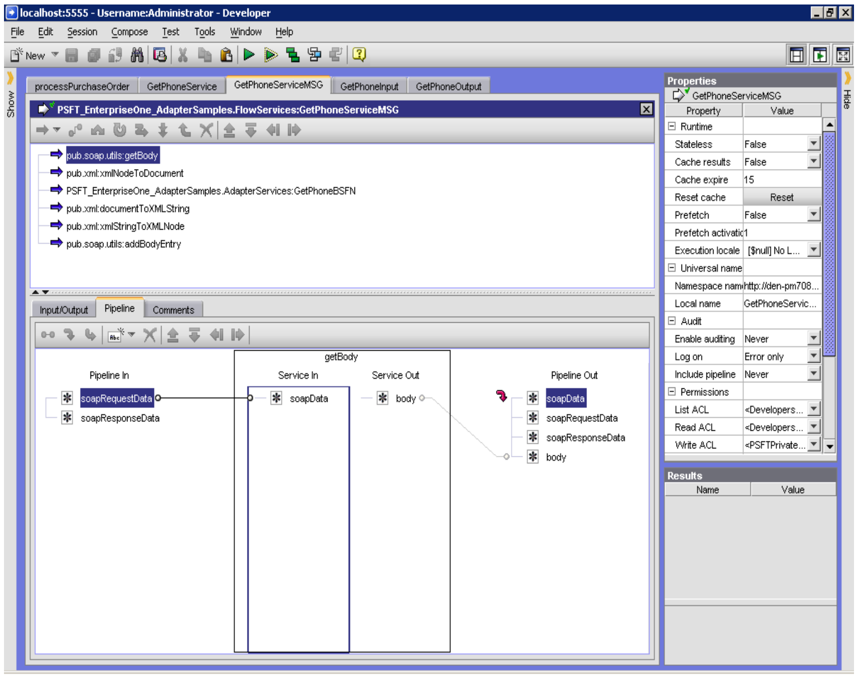
PSFT_EnterpriseOne_AdapterSamples.BusinessFunction.docs:GetPhoneOutput

```
wm:GetPhoneServiceMSG
  GetPhone
    Request
      AddressNumber
    Response
      AddressNumber
      LineNumber
      AreaCode
      PhoneNumber
      PhoneNumberType
```

Properties: wm:GetPhoneServiceMSG

Property	Value
General	
Name	wm:GetPhoneServiceMSG
Type	Document
XML namespace	http://den-pm7082504.peoplesoft.com
Comments	
String display type	Text Field
Pick list choices	
Document reference	
Constraints	
Required	True
Allow null	True
Allow unspecified fields	False
Content type	
Java wrapper type	UNKNOWN

Next, create the service flow. Double-click on the flow service in the navigation pane to open the service editor. Click on Insert and choose Browse. Locate WmPublic package. Navigate to Pub/Soap/Utils. Choose getBody method. Click on inserted step. Choose Pipeline tab in lower frame and map input/output as follows:



localhost:5555 - Username:Administrator - Developer

PSFT_EnterpriseOne_AdapterSamples.FlowServices:GetPhoneServiceMSG

```
pub.soap.utils:getBody
pub.xml:xmlNodeToDocument
PSFT_EnterpriseOne_AdapterSamples.AdapterServices:GetPhoneBSFN
pub.xml:documentToXMLString
pub.xml:xmlStringToXMLNode
pub.soap.utils:addBodyEntry
```

Pipeline (getBody):

Pipeline In	Service In	Service Out	Pipeline Out
soapRequestData	soapData	body	soapData
soapResponseData			soapRequestData
			soapResponseData
			body

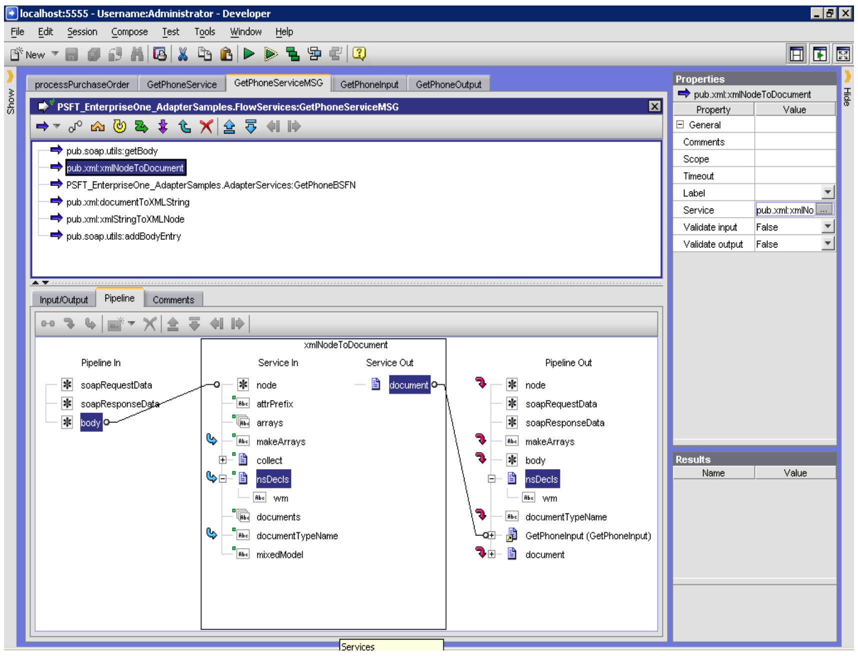
Properties: GetPhoneServiceMSG

Property	Value
Runtime	
Stateless	False
Cache results	False
Cache expire	15
Reset cache	Reset
Prefetch	False
Prefetch activatio	1
Execution locale	[$null] No L...
Universal name	
Namespace nam	http://den-pm708...
Local name	GetPhoneServic...
Audit	
Enable auditing	Never
Log on	Error only
Include pipeline	Never
Permissions	
List ACL	<Developers...
Read ACL	<Developers...
Write ACL	<PSFTPrivate...

- Map soapRequestData from Pipeline In to soapData in Service In
- Map body from Service Out to body in Pipeline Out.
- Drop soapData object from Pipeline Out.

To insert next step, click on Insert and Browse. Choose xmlNodeToDocument method from WmPublic/Pub/xml.

Click on pub.xml:xmlNodeToDocument step in upper frame and click on Pipeline tab in lower frame to map input/output parameters for the step. Map the parameters as follows:


localhost:5555 - Username:Administrator - Developer

PSFT_EnterpriseOne_AdapterSamples.FlowServices:GetPhoneServiceMSG

```
pub.soap.utils:getBody
pub.xml:xmlNodeToDocument
PSFT_EnterpriseOne_AdapterSamples.AdapterServices:GetPhoneBSFN
pub.xml:documentToXMLString
pub.xml:xmlStringToXMLNode
pub.soap.utils:addBodyEntry
```

Properties: pub.xml:xmlNodeToDocument

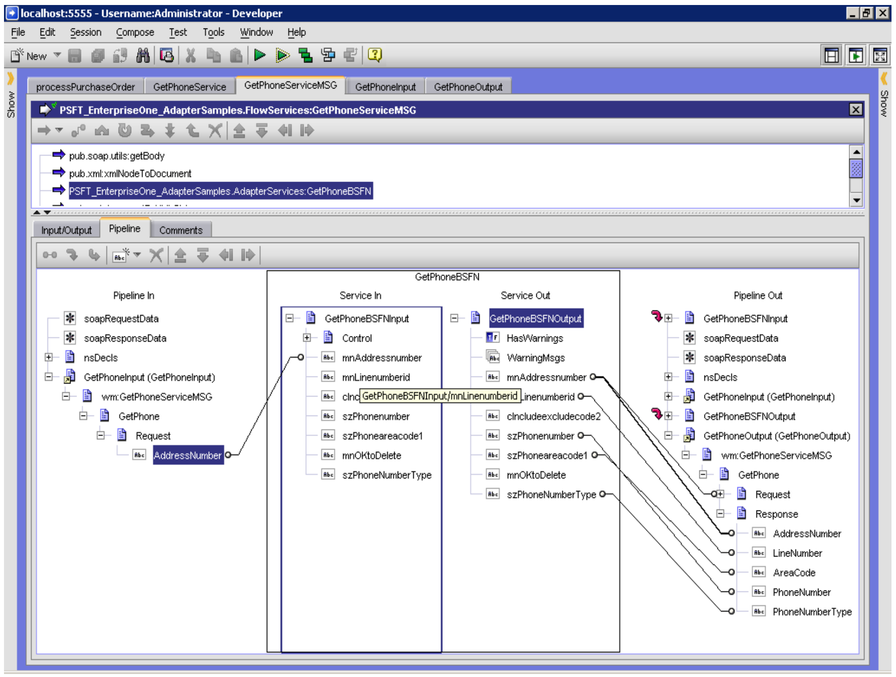
Property	Value
General	
Comments	
Scope	
Timeout	
Label	
Service	pub.xml:xmlNo
Validate input	False
Validate output	False

Map body from Pipeline In to node in Service In. Choose makeArrays in ServiceIn and click on Set Value. Set the value to False. Choose nsDecls and click on Set Value. In the set value dialog, click on Add Row button on extreme right top corner. Put name as the namespace name you selected for document type. Put Value as XML namespace value of the service. Both of these values come from step 8.a above.

Choose documentTypeName and click Set Value. Set the value as fully qualified name of the input document type created as in the Example: PSFT_BPEL_PM_Tests.Docs:GetPhoneInput). Click on Pipeline Out frame. Do the same setting for nsDecls.

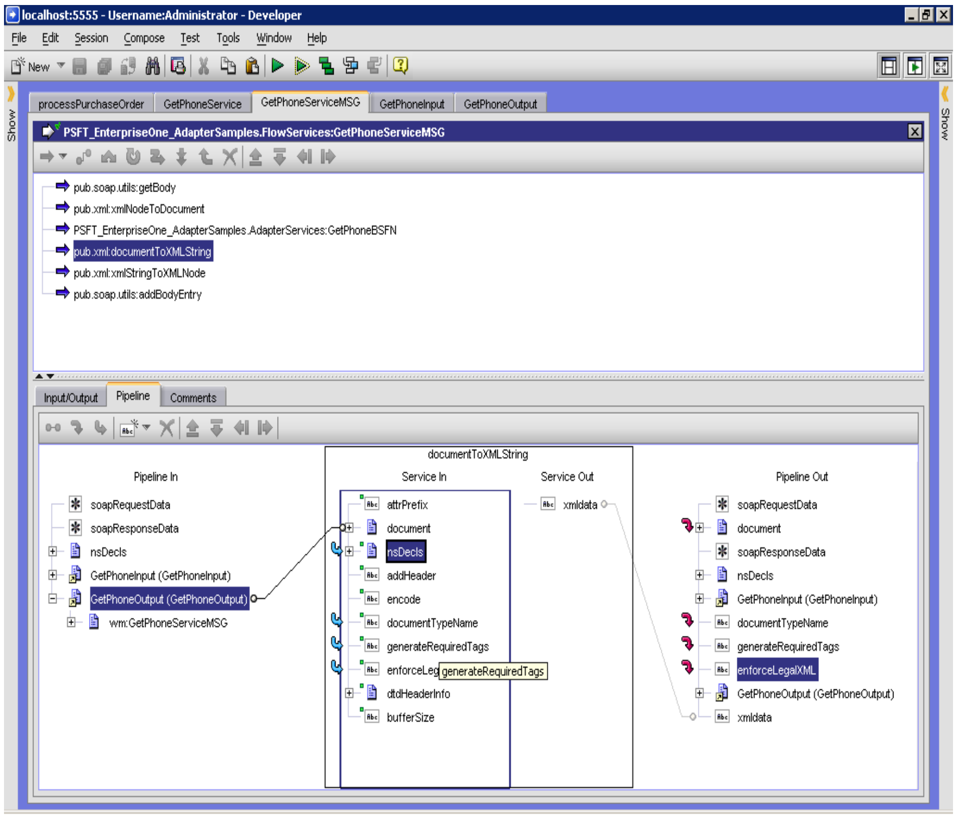
Insert new Document Reference variable and choose Input Document type, created in step 8, in the dialog box that appears. Give it your desired name.Map document from Service Out to Document Reference just created in Pipeline Out. Drop node, makeArrays, body, documentTypeName and document objects from Pipeline Out.

Insert the BSFN Adapter service that will call EnterpriseOne BSFN for this task. Click on the inserted step and choose Pipeline tab from the lower frame to map input/output parameters as follows:



- Map input parameters from Input Document reference in Pipeline In to appropriate input parameters in Service In.
- Click on Pipeline Out frame. Add new Document reference and choose the output document type created in step 9. Give it appropriate name.
- Map output parameters from Service Out to appropriate parameters under output document reference in Pipeline Out.
- Drop Input and Output document type that are in Service In and Service out respectively and appear in Pipeline Out (see screen shot above).

For reverse conversion, insert documentToXMLString method from WmPublic/pub/xml. Click on the step and Pipeline tab in lower frame to set input/output parameters as follows:



PSFT_EnterpriseOne_AdapterSamples.FlowServices:GetPhoneServiceMSG

- pub.soap.utils:getBody
- pub.xml:xmlNodeToDocument
- PSFT_EnterpriseOne_AdapterSamples.AdapterServices:GetPhoneBSFN
- pub.xml:documentToXMLString
- pub.xml:xmlStringToXMLNode
- pub.soap.utils:addBodyEntry

documentToXMLString

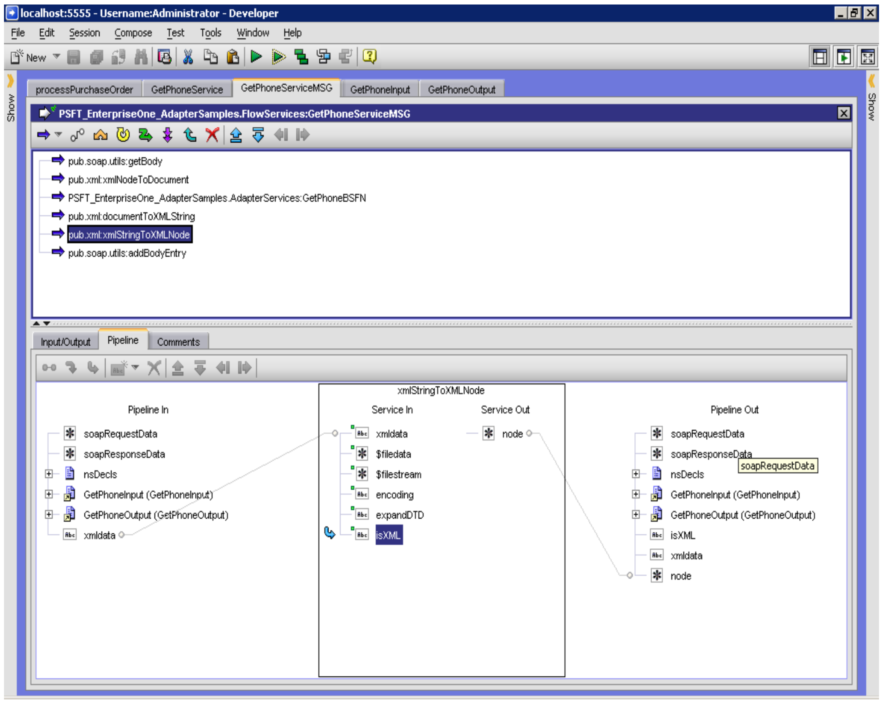
Pipeline In: soapRequestData, soapResponseData, nsDecls, GetPhoneInput (GetPhoneInput), GetPhoneOutput (GetPhoneOutput), wm:GetPhoneServiceMSG

Service In: attrPrefix, document, nsDecls, addHeader, encode, documentTypeName, generateRequiredTags, enforceLeg generateRequiredTags, dtdHeaderInfo, bufferSize

Service Out: xmldata

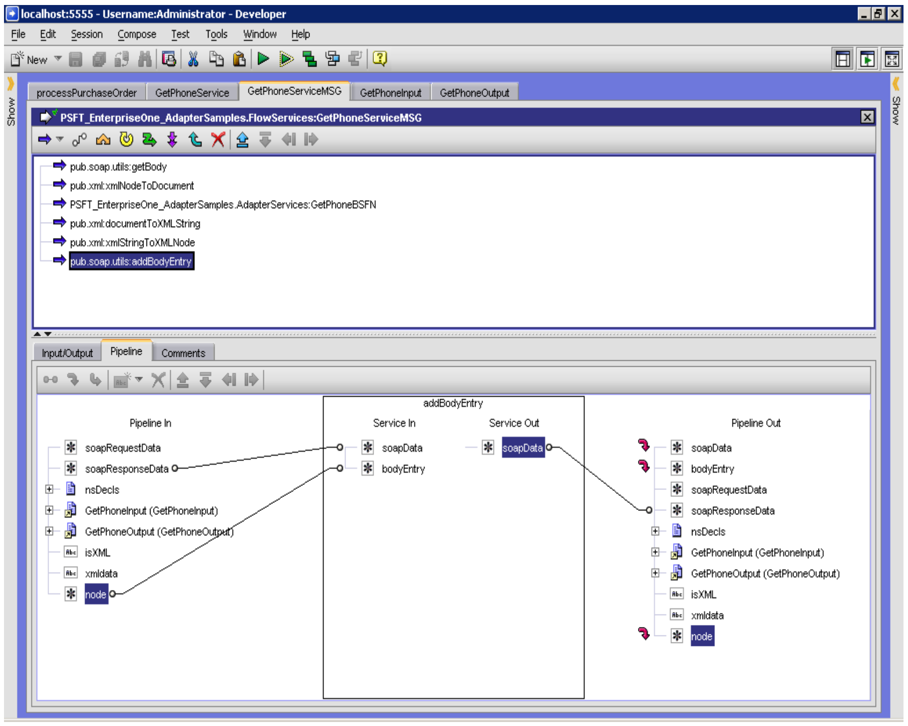
Pipeline Out: soapRequestData, document, soapResponseData, nsDecls, GetPhoneInput (GetPhoneInput), documentTypeName, generateRequiredTags, enforceLegalXML, GetPhoneOutput (GetPhoneOutput), xmldata

- Map the output document reference from Pipeline In to document element in Service In.
- Set the value for nsDecls as in step 15.c above.
- Set the value of documentTypeName element to the fully qualified output document type created in step 9. Example: PSFT_BPEL_PM_Tests.Docs:GetPhoneOutput
- Set value of generateRequiredTags to True.
- Set value of enforceLegalXMl to True.
- Map xmldata from Service Out to xmldata element in Piepline Out.
- Drop document, documentTypeName, generateRequiredTagd and enforceLegalXML element from Pipeline Out.
- Next insert xmlStringToXMLNode method from WmPublic/pub/xml. Set the pipeline parameters as follows:


localhost:5555 - Username:Administrator - Developer
File Edit Session Compose Test Tools Window Help
processPurchaseOrder GetPhoneService GetPhoneServiceMSG GetPhoneInput GetPhoneOutput
PSFT_EnterpriseOne_AdapterSamples.FlowServices:GetPhoneServiceMSG
pub.soap.utils:getBody
pub.xml:xmlNodeToDocument
PSFT_EnterpriseOne_AdapterSamples.AdapterServices:GetPhoneBSFN
pub.xml:documentToXMLString
pub.xml:xmlStringToXMLNode
pub.soap.utils:addBodyEntry
Input/Output Pipeline Comments
xmlStringToXMLNode
Pipeline In
Service In
Service Out
Pipeline Out
soapRequestData
soapResponseData
nsDecls
GetPhoneInput (GetPhoneInput)
GetPhoneOutput (GetPhoneOutput)
xmldata
$filedata
$filestream
encoding
expandDTD
isXML
node
soapRequestData

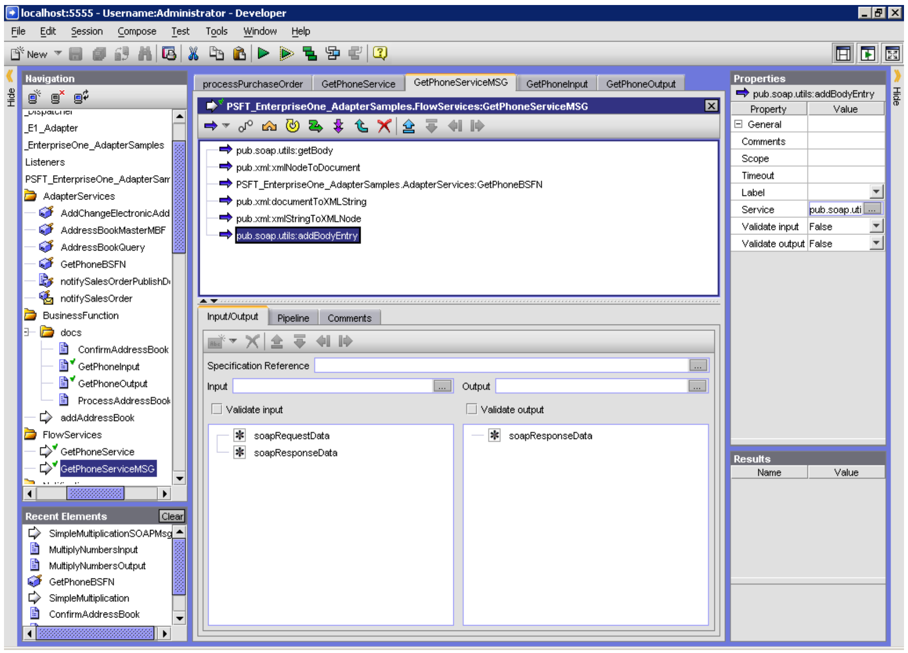
- Map xmldata from Pipeline In to xmldata in Service In.
- Set value of is XML to True.
- Map node from Service Out to Pipeline Out.

Insert addBodyEntry method from WmPublic/pub/soap/utils. Map parameters as follows:


localhost:5555 - Username:Administrator - Developer
File Edit Session Compose Test Tools Window Help
New
processPurchaseOrder
GetPhoneService
GetPhoneServiceMSG
GetPhoneInput
GetPhoneOutput
PSFT_EnterpriseOne_AdapterSamples.FlowServices:GetPhoneServiceMSG
pub.soap.utils:getBody
pub.xml:xmlNodeToDocument
PSFT_EnterpriseOne_AdapterSamples.AdapterServices:GetPhoneBSFN
pub.xml:documentToXMLString
pub.xml:xmlStringToXMLNode
pub.soap.utils:addBodyEntry
Input/Output
Pipeline
Comments
addBodyEntry
Pipeline In
soapRequestData
soapResponseData
nsDecls
GetPhoneInput (GetPhoneInput)
GetPhoneOutput (GetPhoneOutput)
isXML
xmldata
node
Service In
soapData
bodyEntry
Service Out
soapData
Pipeline Out
soapData
bodyEntry
soapRequestData
soapResponseData
nsDecls
GetPhoneInput (GetPhoneInput)
GetPhoneOutput (GetPhoneOutput)
isXML
xmldata
node
Show

- Map soapResponseData in Pipeline In to soapData in Service In.
- Map node in Pipeline In to bodyEntry in Service In.
- Map soapData in Service Out to soapResponseData in Pipeline Out.
- Drop soapData, bodyEntry and node from Pipeline Out.

Service is now ready. Save all the work.


localhost:5555 - Username:Administrator - Developer
File Edit Session Compose Test Tools Window Help
New
Navigation
Hide

_Dispatcher
_E1_Adapter
_EnterpriseOne_AdapterSamples
Listeners
PSFT_EnterpriseOne_AdapterSam
AdapterServices
AddChangeElectronicAdd
AddressBookMasterMBF
AddressBookQuery
GetPhoneBSFN
notifySalesOrderPublishD
notifySalesOrder
BusinessFunction
docs
ConfirmAddressBook
GetPhoneInput
GetPhoneOutput
ProcessAddressBook
addAddressBook
FlowServices
GetPhoneService
GetPhoneServiceMSG

Recent Elements
SimpleMultiplicationSOAPMsg
MultiplyNumbersInput
MultiplyNumbersOutput
GetPhoneBSFN
SimpleMultiplication
ConfirmAddressBook

processPurchaseOrder | GetPhoneService | GetPhoneServiceMSG | GetPhoneInput | GetPhoneOutput

PSFT_EnterpriseOne_AdapterSamples.FlowServices:GetPhoneServiceMSG

```
pub.soap.utils:getBody
pub.xml:xmlNodeToDocument
PSFT_EnterpriseOne_AdapterSamples.AdapterServices:GetPhoneBSFN
pub.xml:documentToXMLString
pub.xml:xmlStringToXMLNode
pub.soap.utils:addBodyEntry
```

Input/Output | Pipeline | Comments

Specification Reference

Input | Output

Validate input | Validate output

soapRequestData
soapResponseData | soapResponseData

Properties

pub.soap.utils:addBodyEntry

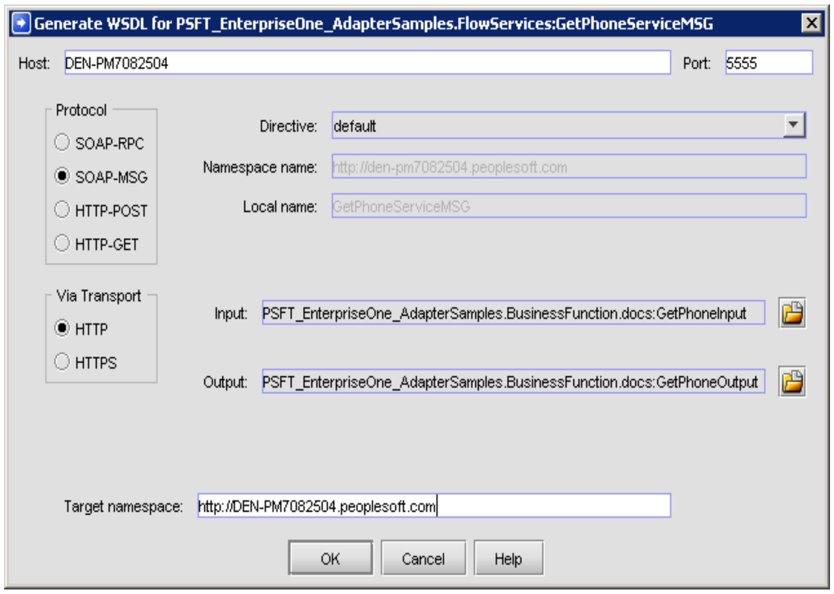
Property	Value
General	
Comments	
Scope	
Timeout	
Label	
Service	pub.soap.uti
Validate input	False
Validate output	False

Results

Name	Value

Generating SOAP-MSG style WSDL

To generate the WSDL for the service, choose Tools – Generate WSDL. Set the value in Generate dialog box as follows:


Generate WSDL for PSFT_EnterpriseOne_AdapterSamples.FlowServices:GetPhoneServiceMSG
OK
Cancel
Help

Host: DEN-PM7082504 Port: 5555

Protocol
- ○ SOAP-RPC
- ● SOAP-MSG
- ○ HTTP-POST
- ○ HTTP-GET

Directive: default

Namespace name: http://den-pm7082504.peoplesoft.com

Local name: GetPhoneServiceMSG

Via Transport
- ● HTTP
- ○ HTTPS

Input: PSFT_EnterpriseOne_AdapterSamples.BusinessFunction.docs:GetPhoneInput

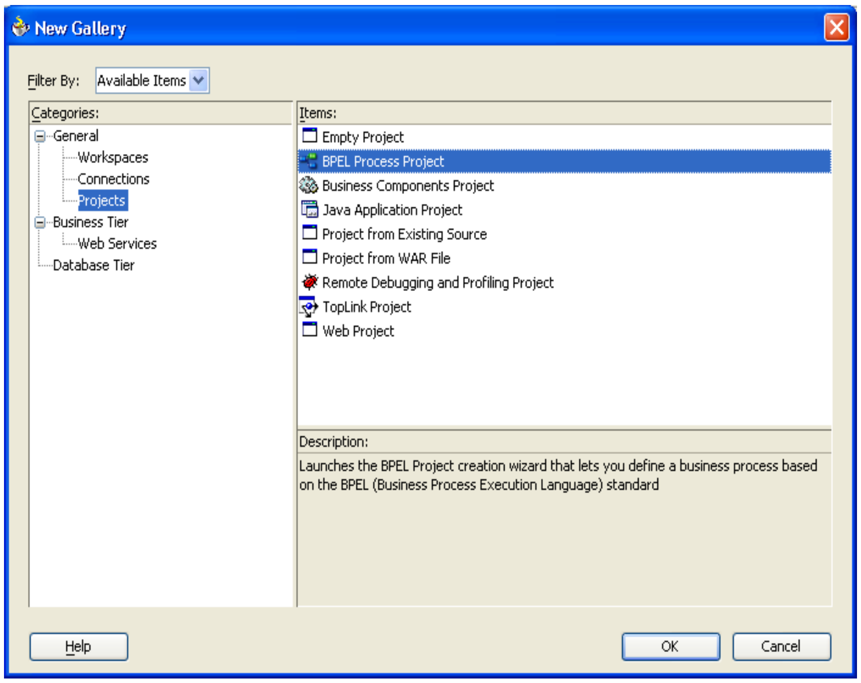
Output: PSFT_EnterpriseOne_AdapterSamples.BusinessFunction.docs:GetPhoneOutput

Target namespace: http://DEN-PM7082504.peoplesoft.com

Set Host name to machine name or IP. Do not use localhost.Set protocol to SOAP-MSG. Directive should remain default. Click on folder icon next to Input to choose Input document type created for the service. Similarly, choose output document style created for the service. Set the value of Target namespace to the XML namespace used for the service. Click OK. Choose the destination folder to save the file. Designer will display message that WSDL was successfully saved.

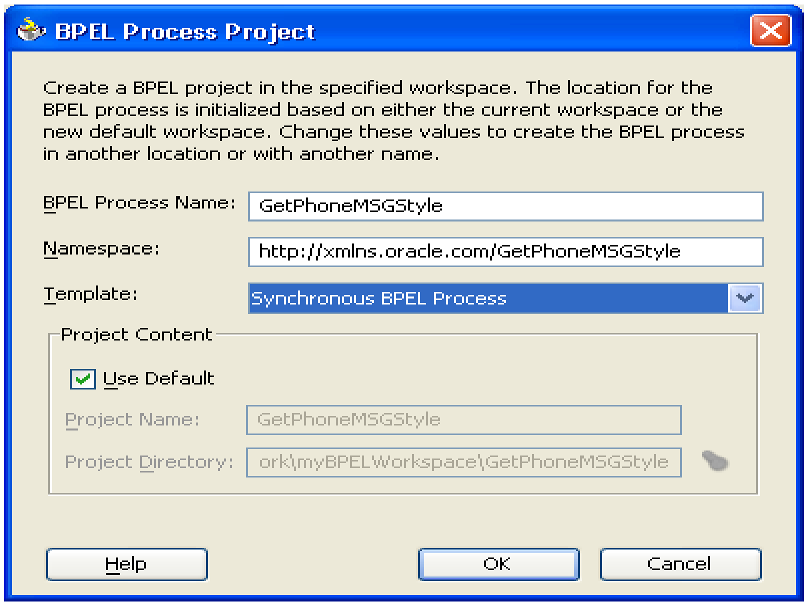
Generating BPEL PM process to consume the service

The following steps assume that BPEL PM server and designer are properly set up and that the appropriate workspace and connections are created in the designer.Start BPEL PM server.Launch BPEL Designer. Right-click on the workspace and choose New Project. From Project Items in right frame choose BPEL Process Project and click OK.



Description:
Launches the BPEL Project creation wizard that lets you define a business process based on the BPEL (Business Process Execution Language) standard

In the project properties dialog, select process name, and choose Synchronous service from Template dropdown. Click OK.



BPEL Process Project

Create a BPEL project in the specified workspace. The location for the BPEL process is initialized based on either the current workspace or the new default workspace. Change these values to create the BPEL process in another location or with another name.

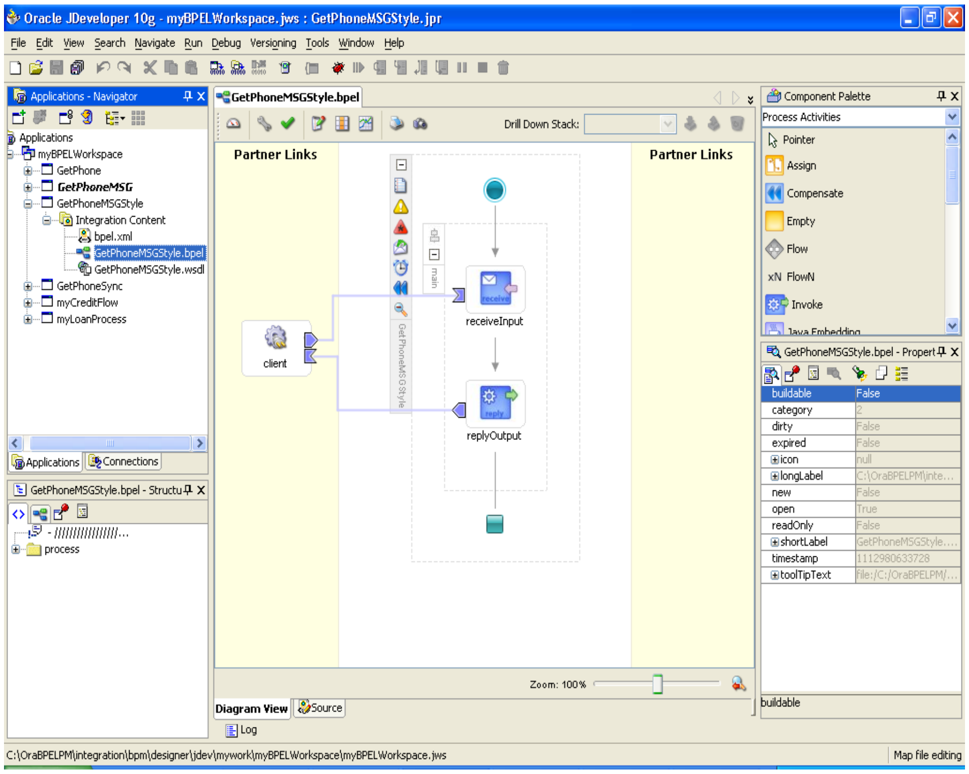
BPEL Process Name:	GetPhoneMSGStyle
Namespace:	http://xmlns.oracle.com/GetPhoneMSGStyle
Template:	Synchronous BPEL Process

Project Content

[✓] Use Default

Project Name:	GetPhoneMSGStyle
Project Directory:	ork\myBPELWorkspace\GetPhoneMSGStyle

The following sections appear. If this does not appear, click Diagram View and double click GetPhoneMsgStyle.bpel in the Applications Navigator section. See the *Oracle BPEL Process Manager Developer's Guide* for a description of these sections.



Oracle JDeveloper 10g - myBPELWorkspace.jws : GetPhoneMSGStyle.jpr

```
Applications
  myBPELWorkspace
    GetPhone
    GetPhoneMSG
    GetPhoneMSGStyle
      Integration Content
        bpel.xml
        GetPhoneMSGStyle.bpel
        GetPhoneMSGStyle.wsdl
    GetPhoneSync
    myCreditFlow
    myLoanProcess
```

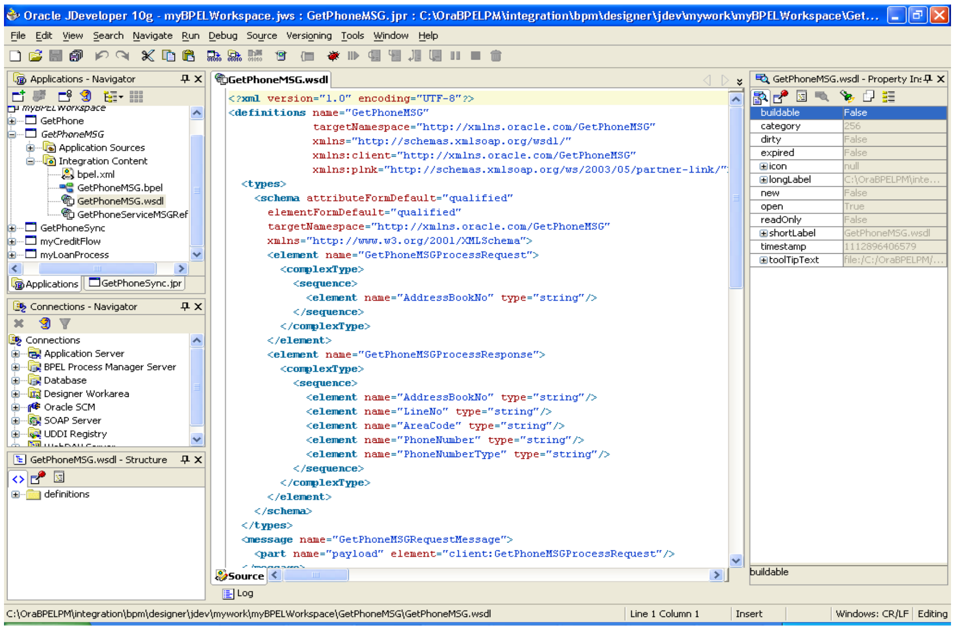
buildable	False
category	2
dirty	False
expired	False
icon	null
longLabel	C:\OraBPELPM\inte...
new	False
open	True
readOnly	False
shortLabel	GetPhoneMSGStyle....
timestamp	1112980633728
toolTipText	file:/C:/OraBPELPM/...

buildable

C:\OraBPELPM\integration\bpm\designer\jdev\mywork\myBPELWorkspace\myBPELWorkspace.jws | Map file editing

Browse to the project directory from explorer and copy the WSDL generated from WSG Developer to this folder.

In Applications –Navigator, navigate to <processname>.WSDL file under Integration Content. Double-click to open the file and view WSDL source.



```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="GetPhoneMSG"
             targetNamespace="http://xmlns.oracle.com/GetPhoneMSG"
             xmlns="http://schemas.xmlsoap.org/wsdl/"
             xmlns:client="http://xmlns.oracle.com/GetPhoneMSG"
             xmlns:plnk="http://schemas.xmlsoap.org/ws/2003/05/partner-link/"
  <types>
    <schema attributeFormDefault="qualified"
      elementFormDefault="qualified"
      targetNamespace="http://xmlns.oracle.com/GetPhoneMSG"
      xmlns="http://www.w3.org/2001/XMLSchema">
      <element name="GetPhoneMSGProcessRequest">
        <complexType>
          <sequence>
            <element name="AddressBookNo" type="string"/>
          </sequence>
        </complexType>
      </element>
      <element name="GetPhoneMSGProcessResponse">
        <complexType>
          <sequence>
            <element name="AddressBookNo" type="string"/>
            <element name="LineNo" type="string"/>
            <element name="AreaCode" type="string"/>
            <element name="PhoneNumber" type="string"/>
            <element name="PhoneNumberType" type="string"/>
          </sequence>
        </complexType>
      </element>
    </schema>
  </types>
  <message name="GetPhoneMSGRequestMessage">
    <part name="payload" element="client:GetPhoneMSGProcessRequest"/>
```

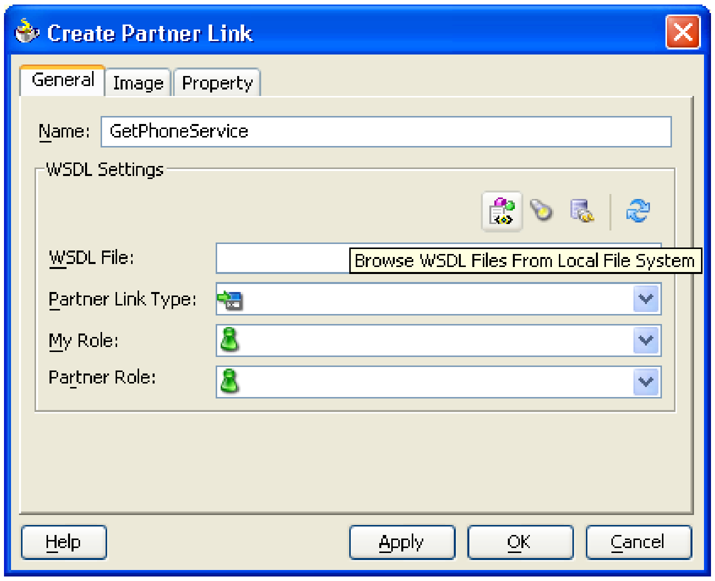
buildable	False
category	256
dirty	False
expired	False
icon	null
longLabel	C:\OraBPELPM\inte...
new	False
open	True
readOnly	False
shortLabel	GetPhoneMSG.wsdl
timestamp	1112896406579
toolTipText	file:/C:/OraBPELPM/...

buildable

C:\OraBPELPM\integration\bpm\designer\jdev\mywork\myBPELWorkspace\GetPhoneMSG\GetPhoneMSG.wsdl | Line 1 Column 1 | Insert | Windows: CR/LF | Editing

Locate the Request and Response elements in the schema. Modify the Request and Response element schema as required for the service. As a reference for this, use elements under Request and Response schema in the copied WSG WSDL file. These elements can be copied from WSG WSDL to process WSDL with minor modification of namespace. Diagram above shows resulting WSDL for this example.

Now double-click on <processname>.bpel file to open it in editor. Right-click on either side of BPEL process (in yellow area) and choose **create partner link** option.



Name: GetPhoneService

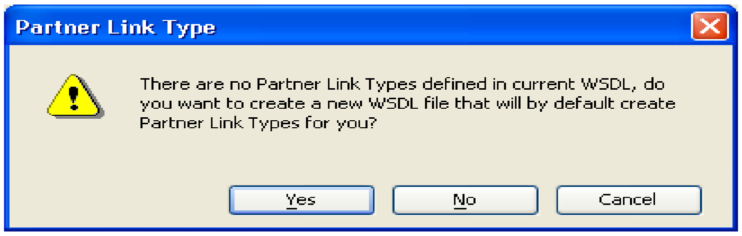
WSDL File:

Partner Link Type:

My Role:

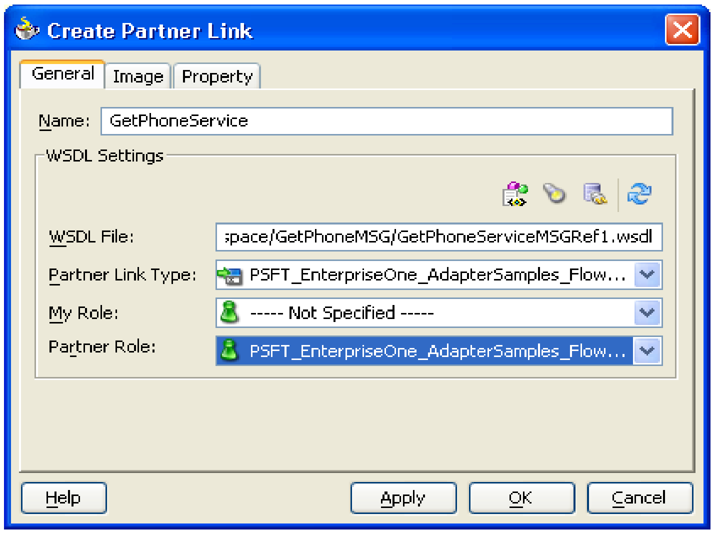
Partner Role:

Provide partner link name. Click on first icon in WSDL Settings frame to select the WSDL file from file system. Select the WSG WSDL copied to the project directory. Designer will display following message:


Partner Link Type

There are no Partner Link Types defined in current WSDL, do you want to create a new WSDL file that will by default create Partner Link Types for you?

Click on Yes to automatically create partner link types. Designer will create partner link type and populate it. Keep My Roles filed as Not specified and populate Partner Role with the value available in drop down.


Create Partner Link

Name: GetPhoneService

WSDL Settings

WSDL File: ;pace/GetPhoneMSG/GetPhoneServiceMSGRef1.wsdl

Partner Link Type: PSFT_EnterpriseOne_AdapterSamples_Flow...

My Role: ----- Not Specified -----

Partner Role: PSFT_EnterpriseOne_AdapterSamples_Flow...

Click OK to close the partner link setting and save bpel file.

Next follow the *BPEL Developer's Guide* to add the following items to the BPEL file.

- Scope Activity
- Invoke construct
- Assign construct

Follow the guide to validate the BPEL file and save the file. To compile and Deploy BPEL process, right-click on project in Application Navigator. Select Deploy > *connection name* > Deploy to Default Domain. Click OK. This will compile and deploy the BPEL process. Watch the bottom of the screen for any errors. Refer to the *BPEL Developer's Guide* for more information.

Notification invoking a BPEL Process

The following steps provide the high level view of how notifications can invoke a BPEL process.

1. Generate a Notification Flow Service using the WSG toolkit from PSFT_EnterpriseOne_AdapterServices.AdatperServices.Notification. package and save the flow.
2. Generate a WSDL for the newly created flow with a SOAP/RPC option and save it as both WSDL and XSD.
3. Create a BPEL process and import the WSDL and the XSD from step 1. Save the process definition.
4. Create WSG Connector.

   Use the WSG connector from the toolkit and open the WSDL saved from step 3. The web service connector will generate a flow service called "initiate", and the corresponding request document type and response document types.
5. Define the Notification trigger in the app designer.

Selecting the EnterpriseOne Notification

Assuming you have installed the latest release of EntepriseOne Integration Points and you have deployed the PSFT_EnterpriseOne_AdapterServices package, you can select one of the already available EnterpriseOne notifications to create your integration.

Prerequisites:

- EnterpriseOne Event/Subscriber setup
- Ensure Transaction Server is running
- Enable E1 Adapter Listener Notification for AdapterServices.Notification.AddressBook:notifyAddressBook (Refer to E1 Adapter Configuration section of WSG Install & Config guide).
- Enable following E1 Adapter Listener: EnterpriseOne_AdapterServices_Listener:EnterpriseOne_Listener (Refer to E1 Adapter Configuration section of WSG Install & Config guide on how to enable Listeners).
- For the purpose of this example we have selected the notifyAddressBook message. The document type definition can be found under the folder AdapterServices.Notification.AddressBook.

Creating a WSG Notification Service

- The notification flow service is responsible for subscribing to the notification, and delivering the notification to BPEL PM by invoking a web service. The flow will be developed in two parts: first defining its interface and generating the web service, and second invoking the connector flow to deliver the notification. Do the following:

Create a new flow service by selecting File > New , then select the Flow Service radio button and click next. Name your new flow notifyAddressBook, select empty flow and click Finish. Assign the notifyAddressBookPublishDocument as the input to the flow.Click the New Variable icon inside the Input/Output panel.

Select Document Reference

Select PSFT_EnterpriseOne_AdapterServices.AdatperServices.Notification.AddressBook.notifyAddressBookPublishDocument. Click Ok. Give this new variable the following name AdapterServices.Notification.AddressBook:notifyAddressBookPublishDocument and Save your flow.

Note: This name has to be exactly the same name as the selected Document Name. You can use copy/paste functionality to ensure this.

Generating the web service interface

Perform the following steps to create the service interface that you will use for the BPEL PM implementation that consumes this notification. Select the Flow you just created with WSG Developer and select Tools > GenerateWSDL. Ensure you have the right host and namespace, select SOAP-RPC and click Ok. Save the file in a known location.

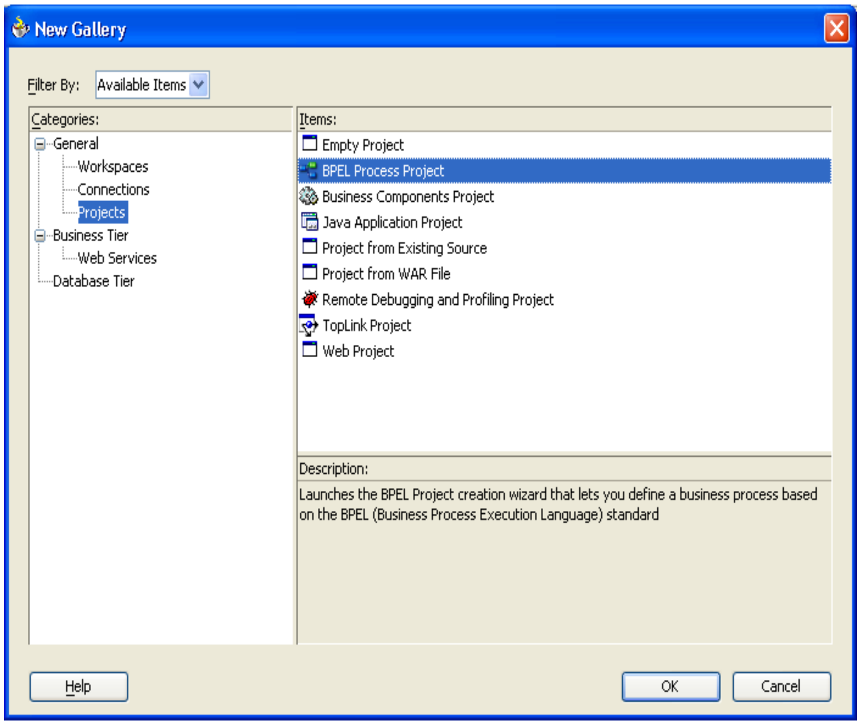
Note: In this case, we have not assigned a universal name for this flow. Keep in mind that the only purpose of this WSDL is to provide a schema definition that can be used when creating the BPEL PM flow that will consume the notification. The WSG flow service we created will be triggered by the notification from EnterpriseOne. This flow is not a web service provider.

After you have generated the WSDL, extract the schema definition from the WSDL into a separate XSD file by copying all the contents inside the <wsdl:types> tags into a separate file. Name this file AddressBook.xsd.

Edit the <xsd:import ..> tag and add the “schemaLocation” attribute. Assign “schemaLocation” attribute the same value as in the “namespace” attribute.
Example: <xsd:import namespace="http://schemas.xmlsoap.org/soap/encoding/" **schemaLocation="http://schemas.xmlsoap.org/soap/encoding/"** />

Creating the BPEL PM Flow

The following steps assume that BPEL PM server and designer are properly set up and that the appropriate workspace and connections are created in the designer. Start BPEL PM server. Launch BPEL Designer. Right-click on the workspace and choose New Project. From Project Items in right frame choose BPEL Process Project and click OK.


New Gallery

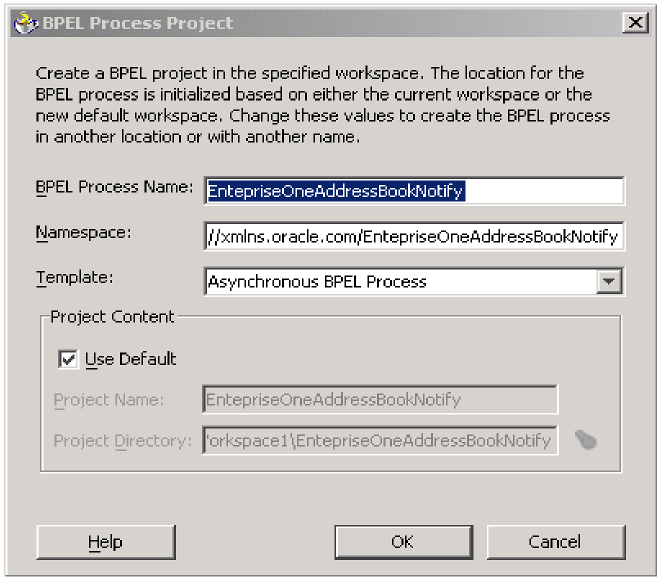
Filter By: Available Items

Categories:
- General
  - Workspaces
  - Connections
  - Projects
- Business Tier
  - Web Services
- Database Tier

Items:
- Empty Project
- BPEL Process Project
- Business Components Project
- Java Application Project
- Project from Existing Source
- Project from WAR File
- Remote Debugging and Profiling Project
- TopLink Project
- Web Project

Description:
Launches the BPEL Project creation wizard that lets you define a business process based on the BPEL (Business Process Execution Language) standard

In the project properties dialog, enter the process name 'EnterpriseOneAddressBookNotify', and choose Asynchronous BPEL Process service from Template dropdown. Click OK.


BPEL Process Project

Create a BPEL project in the specified workspace. The location for the BPEL process is initialized based on either the current workspace or the new default workspace. Change these values to create the BPEL process in another location or with another name.

BPEL Process Name: EntepriseOneAddressBookNotify

Namespace: //xmlns.oracle.com/EntepriseOneAddressBookNotify

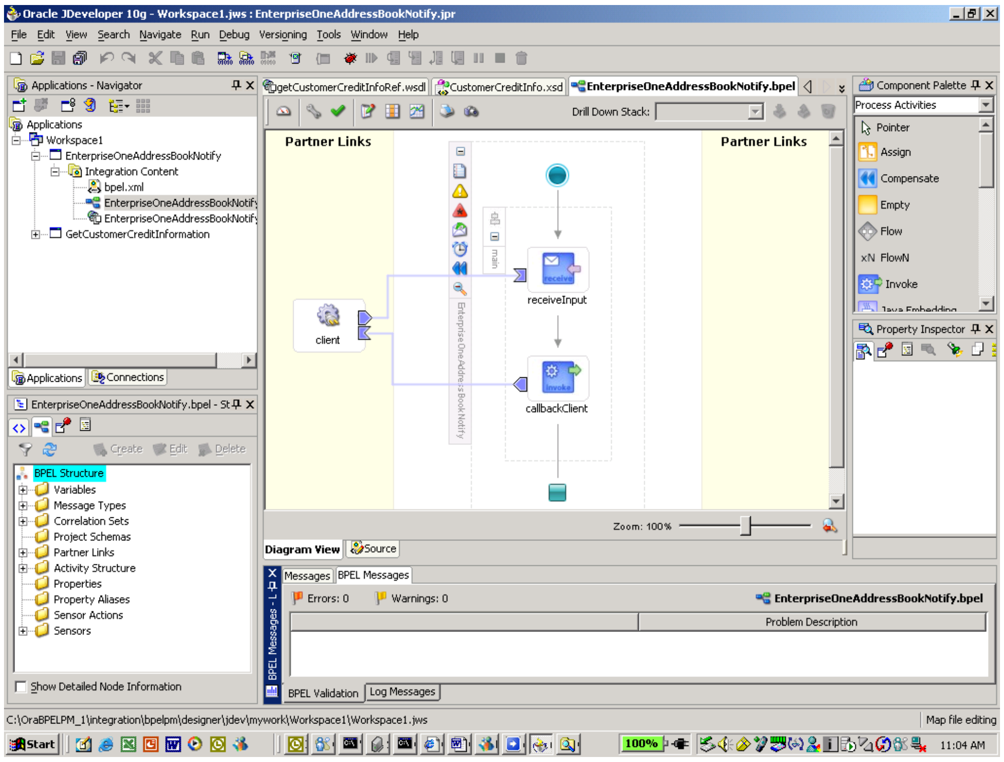
Template: Asynchronous BPEL Process

Project Content

[✓] Use Default

Project Name: EntepriseOneAddressBookNotify

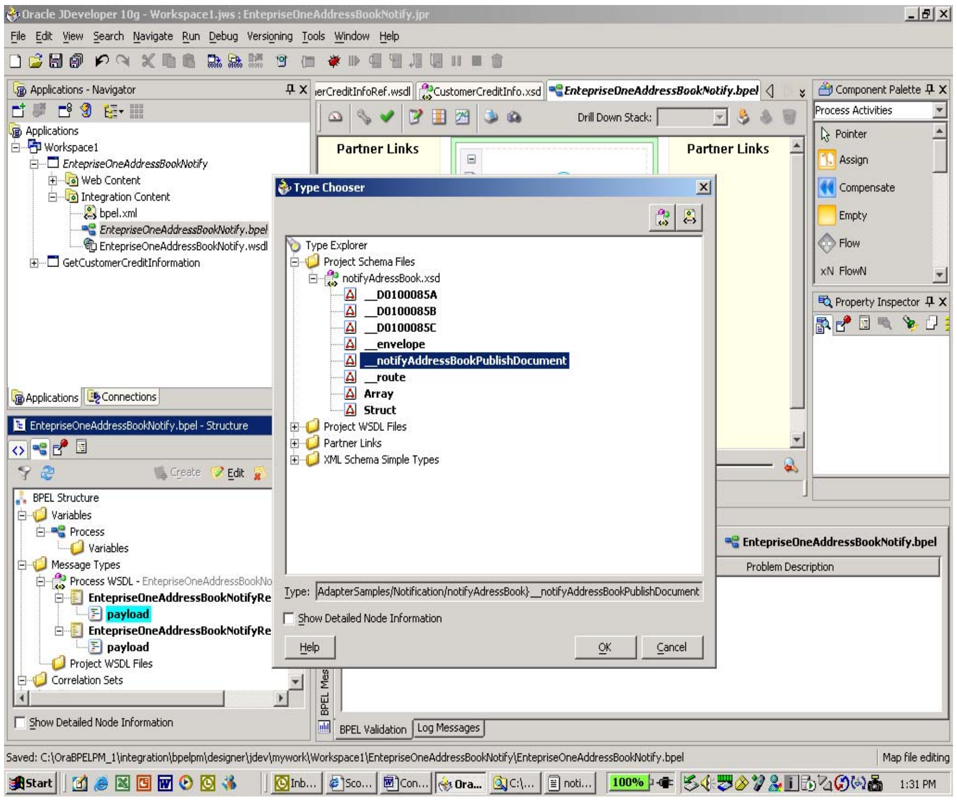
Project Directory: 'orkspace1\EntepriseOneAddressBookNotify



Browse to the project directory from explorer and copy the XSD file that you created using WSG Developer to this project folder.

Import the AdressBook Schema into the project. In the BPEL Structure window, right click on Project Schemas and select Import Schemas, in the Import Schema window, browse the file system and select the AdressBook.xsd. Select Open, ok, and save all.

Associate Schema to Request Message type. In the BPEL Structure window, select Message Types > Process WSDL > EnterpriseOneAddressBookNotifyRequestMessage > payload, right click on payload and select Edit Message Part. Select the type radio button, and click the browse types button. In the Type Chooser window, select Project Schema Files > AddressBook.xsd > __notifyAddressBookPublishDocument. Click ok, ok, and save all.



Type Chooser

- Type Explorer
  - Project Schema Files
    - notifyAdressBook.xsd
      - __D0100085A
      - __D0100085B
      - __D0100085C
      - __envelope
      - __notifyAddressBookPublishDocument
      - __route
      - Array
      - Struct
  - Project WSDL Files
  - Partner Links
  - XML Schema Simple Types

Type: AdapterSamples/Notification/notifyAdressBook}__notifyAddressBookPublishDocument

Show Detailed Node Information

Help OK Cancel

- Applications
  - Workspace1
    - EnteipriseOneAddressBookNotify
      - Web Content
      - Integration Content
        - bpel.xml
        - EnteipriseOneAddressBookNotify.bpel
        - EnteipriseOneAddressBookNotify.wsdl
    - GetCustomerCreditInformation

EnteipriseOneAddressBookNotify.bpel - Structure

- BPEL Structure
  - Variables
    - Process
      - Variables
  - Message Types
    - Process WSDL - EnteipriseOneAddressBookNo
      - EnteipriseOneAddressBookNotifyRe
        - payload
      - EnteipriseOneAddressBookNotifyRe
        - payload
    - Project WSDL Files
  - Correlation Sets

Saved: C:\OraBPELPM_1\integration\bpelpm\designer\jdev\mywork\Workspace1\EnteipriseOneAddressBookNotify\EnteipriseOneAddressBookNotify.bpel

Compile and Deploy the process. Right-click on EnterpriseOneAddressBookNotify.bpel node in the Application Navigator. Select Deploy > LocalBPELServer > Deploy to default domain

Generating the WSG web service Connector

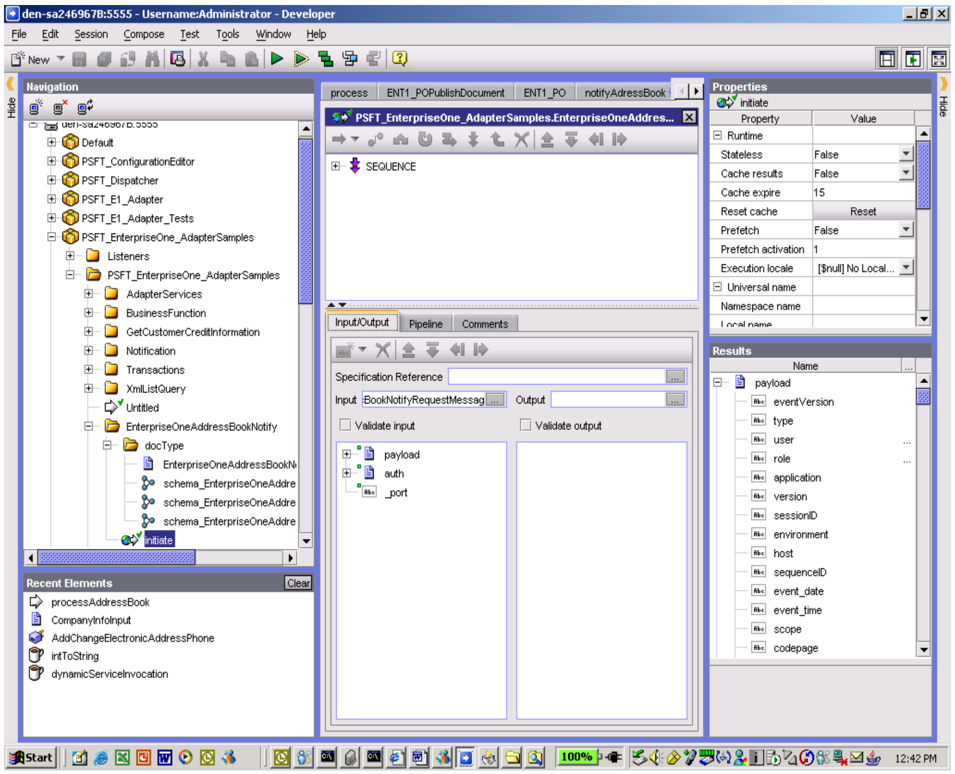
Follow these steps to create the WSG web service connector:

1. Copy WSDL from BPEL PM Server to a WSG developer location.
2. Open the BPEL Console for your active BPEL Server.
3. Select the EnterpriseOneAddressBookNotify link.
4. Copy the URL of the WSDL location.

Create WSG Connector

1. Start WSG Developer and connect to an active WSG Server.
2. Select File > New, choose the Web Service Connector, and click Next.
3. Select the WSG Folder under which to create the connector and click Next.
4. Paste the URL of WSDL (from step 1c) and Click Open.
5. Click Finish.

A web service connector will be generated automatically with a flow service called “initiate”, and the corresponding request document type.



Specification Reference

Input BookNotifyRequestMessag

Output

Validate input

Validate output

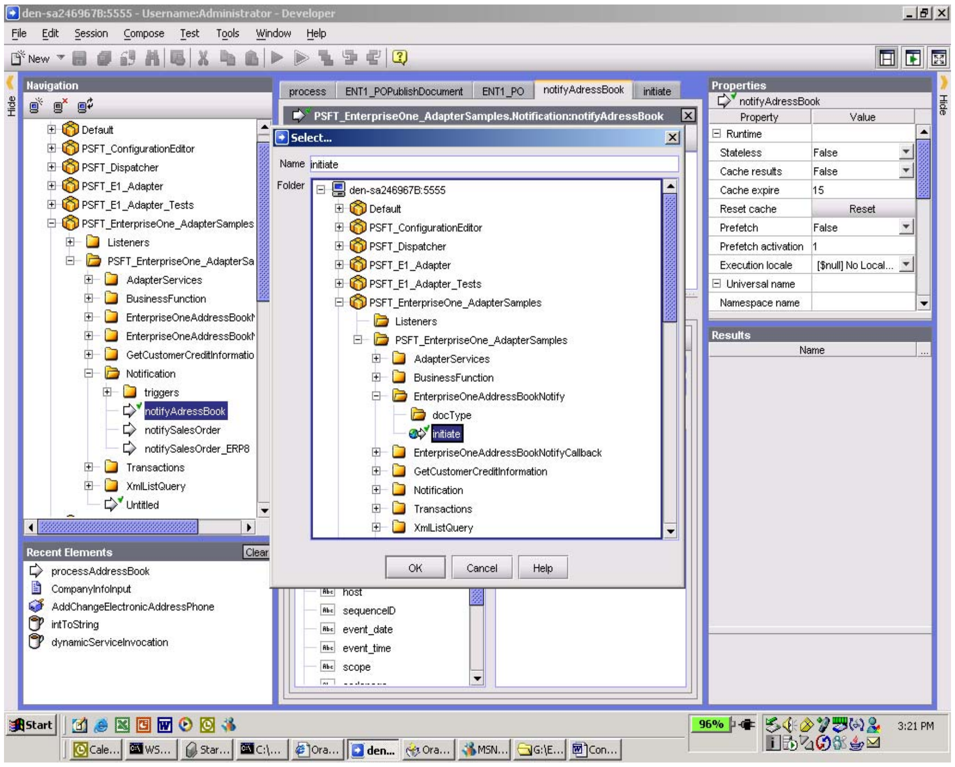
Property	Value
Runtime	
Stateless	False
Cache results	False
Cache expire	15
Reset cache	Reset
Prefetch	False
Prefetch activation	1
Execution locale	[$null] No Local...
Universal name	
Namespace name	

Name
- payload
  - eventVersion
  - type
  - user
  - role
  - application
  - version
  - sessionID
  - environment
  - host
  - sequenceID
  - event_date
  - event_time
  - scope
  - codepage

Updating the Notification flow to deliver the notification

Go through the following steps to update your notification flow so that it calls the new connector flow that you created to deliver the notification message to the BPEL PM process:

1. In WSG Developer, select the notifyAddressBook flow service you created in the first part of this exercise.
2. Insert an invoke step by clicking on the Insert icon and select Browse.
3. Within the Select window navigate to the web service connector flow and click ok.
4. Select the invoke step, and click on the Pipeline Tab. There is no document under payload. Map the input document to the payload structure.
5. Save your work.


den-sa246967B:5555 - Username:Administrator - Developer
File Edit Session Compose Test Tools Window Help
New
Navigation
process ENT1_POPublishDocument ENT1_PO notifyAdressBook initiate
PSFT_EnterpriseOne_AdapterSamples.Notification:notifyAdressBook
Properties
Results
Recent Elements
Clear

Select...

Name: initiate

Folder:
- den-sa246967B:5555
  - Default
  - PSFT_ConfigurationEditor
  - PSFT_Dispatcher
  - PSFT_E1_Adapter
  - PSFT_E1_Adapter_Tests
  - PSFT_EnterpriseOne_AdapterSamples
    - Listeners
    - PSFT_EnterpriseOne_AdapterSamples
      - AdapterServices
      - BusinessFunction
      - EnterpriseOneAddressBookNotify
        - docType
        - initiate
      - EnterpriseOneAddressBookNotifyCallback
      - GetCustomerCreditInformation
      - Notification
      - Transactions
      - XmlListQuery

OK Cancel Help

Navigation: Default, PSFT_ConfigurationEditor, PSFT_Dispatcher, PSFT_E1_Adapter, PSFT_E1_Adapter_Tests, PSFT_EnterpriseOne_AdapterSamples, Listeners, PSFT_EnterpriseOne_AdapterSa, AdapterServices, BusinessFunction, EnterpriseOneAddressBookN, EnterpriseOneAddressBookN, GetCustomerCreditInformatio, Notification, triggers, notifyAdressBook, notifySalesOrder, notifySalesOrder_ERP8, Transactions, XmlListQuery, Untitled

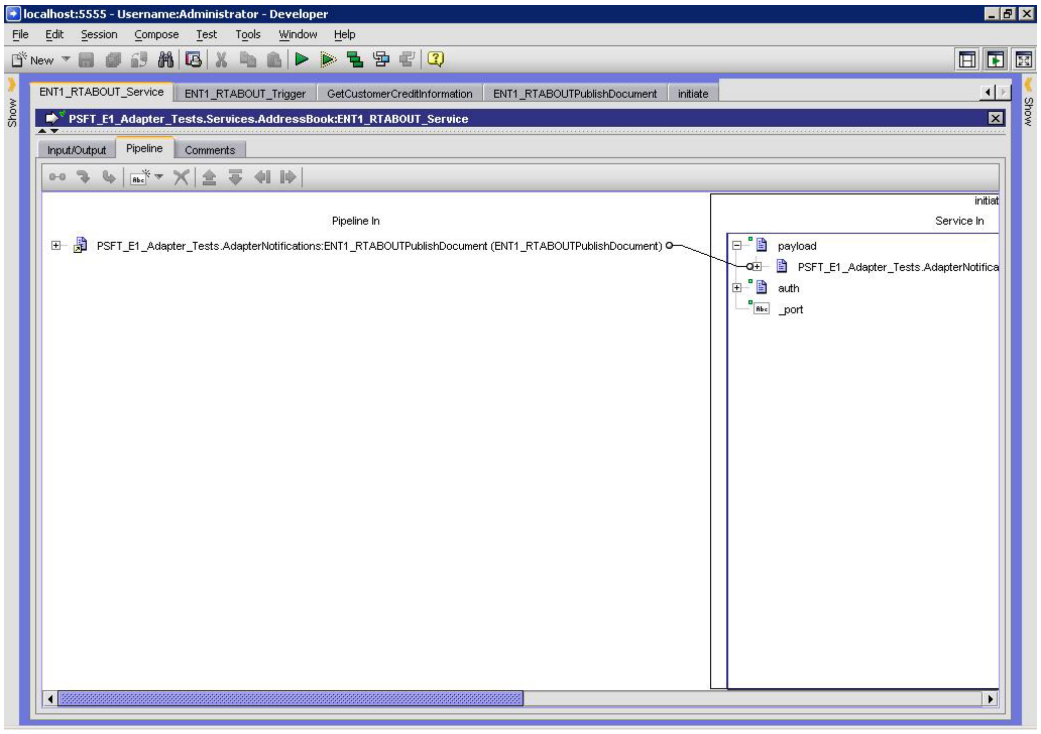
Recent Elements: processAddressBook, CompanyInfoInput, AddChangeElectronicAddressPhone, intToString, dynamicServiceInvocation

Pipeline: host, sequenceID, event_date, event_time, scope

Properties: notifyAdressBook

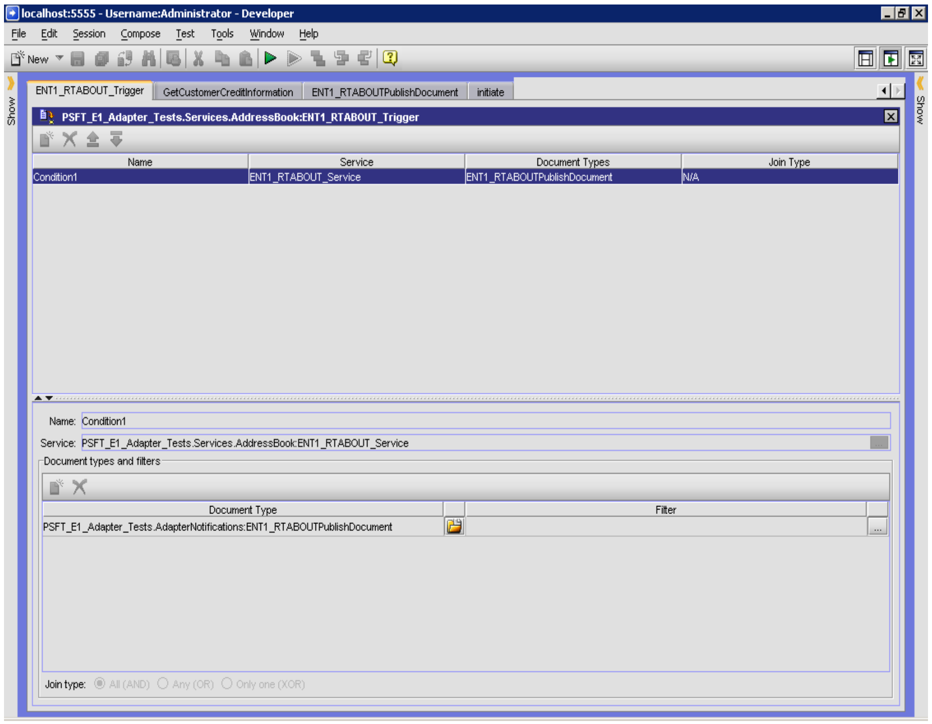
Property	Value
Runtime	
Stateless	False
Cache results	False
Cache expire	15
Reset cache	Reset
Prefetch	False
Prefetch activation	1
Execution locale	[$null] No Local...
Universal name	
Namespace name	

Results: Name


localhost:5555 - Username:Administrator - Developer
File Edit Session Compose Test Tools Window Help
New
ENT1_RTABOUT_Service
ENT1_RTABOUT_Trigger
GetCustomerCreditInformation
ENT1_RTABOUTPublishDocument
initiate
PSFT_E1_Adapter_Tests.Services.AddressBook:ENT1_RTABOUT_Service
Input/Output
Pipeline
Comments
Pipeline In
PSFT_E1_Adapter_Tests.AdapterNotifications:ENT1_RTABOUTPublishDocument (ENT1_RTABOUTPublishDocument)
initiat
Service In
payload
PSFT_E1_Adapter_Tests.AdapterNotifica
auth
_port
Show
Show

Defining the Notification Trigger

Create a new Trigger by selecting File > New, and choosing the Trigger radio button. Click Next. Name your Trigger AddressBookNotify_Trigger and click Finish. In the Service field in the bottom panel click on Select a service and choose the notification service you just created. Click on Insert document types icon and choose PSFT_EnterpriseOne_AdapterServices.AdatperServices.Notification.AddressBook.notifyAddressBookPublishDocument and Save your work


localhost:5555 - Username:Administrator - Developer
File Edit Session Compose Test Tools Window Help
New
ENT1_RTABOUT_Trigger
GetCustomerCreditInformation
ENT1_RTABOUTPublishDocument
initiate
Show
PSFT_E1_Adapter_Tests.Services.AddressBook:ENT1_RTABOUT_Trigger

Name	Service	Document Types	Join Type
Condition1	ENT1_RTABOUT_Service	ENT1_RTABOUTPublishDocument	N/A

Name: Condition1

Service: PSFT_E1_Adapter_Tests.Services.AddressBook:ENT1_RTABOUT_Service

Document types and filters

Document Type	Filter
PSFT_E1_Adapter_Tests.AdapterNotifications:ENT1_RTABOUTPublishDocument	

Join type: [✓ All (AND)] Any (OR) Only one (XOR)

•

WSG initiating a BPEL Process

Selecting desired BPEL Process

Create a BPEL Process and Copy WSDL from BPEL PM Server to a WSG developer location.

1. Open the BPEL Console for your active BPEL Server.
2. Click on the desired process link.
3. Click on WSDL and then on the WSDL Location link.
4. Save the WSDL in the WSG directory (make sure to change the extension to .wsdl).

You should be able to select any BPEL PM provided service to consume from WSG. For the purpose of this example, we will take the process that was created in the section titled "BPEL Consuming EnterpriseOne IP web service using SOAP/RPC." This BPEL PM process invokes a service back in WSG. We will call this BPEL PM process from a WSG connector for demonstrating this feature.

Creating the WSG Service Connector

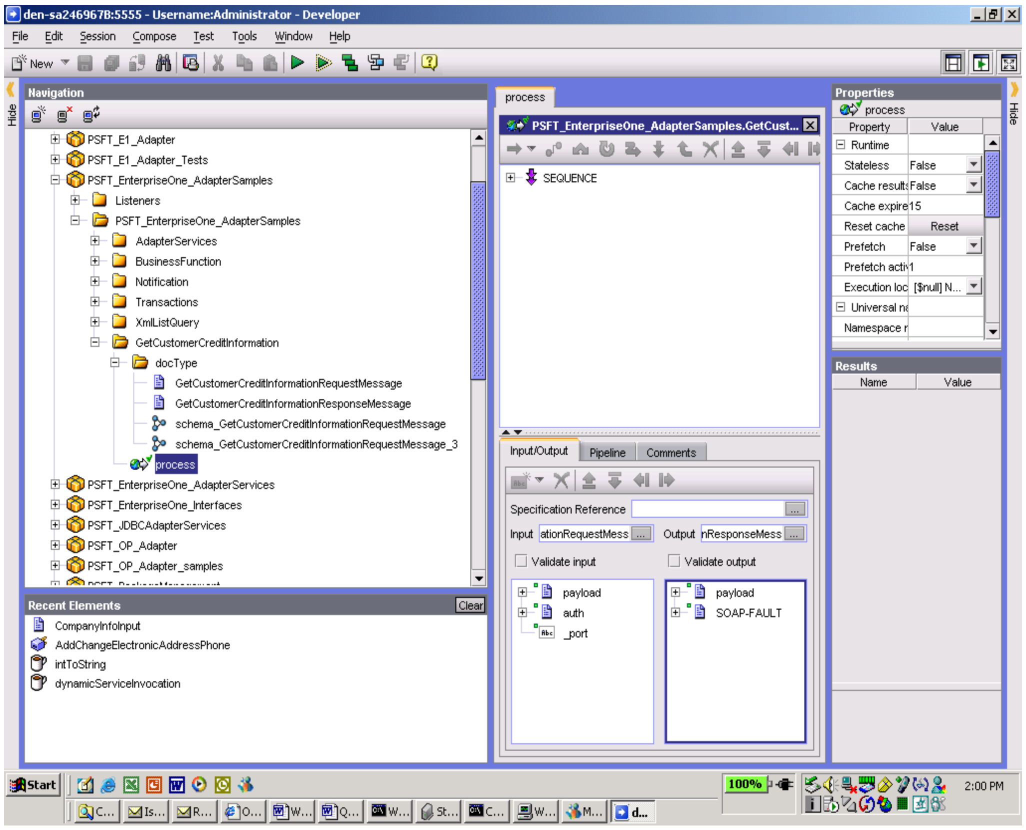
Assuming you have created and deployed the GetCustomerCreditInfo project to the Oracle BPEL PM Server, follow these steps:

1. Copy WSDL from BPEL PM Server to a WSG developer location.
2. Open the BPEL Console for your active BPEL Server.
3. Click on the GetCustomerCreditInformation link.
4. Click on WSDL and then on the WSDL Location link.
5. Go to File > Save As and save the WSDL in the WSG directory (make sure to change the extension to .wsdl).

Create WSG Connector

1. Start WSG Developer and connect to an active WSG Server.
2. Select File > New, choose the Web Service Connector, and click Next.
3. Select the WSG Folder under which to create the connector and click Next.
4. Click the Pick icon to find the WSDL file you copied in step 1.
5. Click Finish.

A web service connector will be generated automatically with a flow service called "process", the corresponding request and response document types.


den-sa246967B:5555 - Username:Administrator - Developer
File Edit Session Compose Test Tools Window Help
New
Hide
Navigation
Recent Elements
Clear
Properties
Results
Input/Output
Pipeline
Comments
Start
100%
2:00 PM

```
PSFT_E1_Adapter
PSFT_E1_Adapter_Tests
PSFT_EnterpriseOne_AdapterSamples
    Listeners
    PSFT_EnterpriseOne_AdapterSamples
        AdapterServices
        BusinessFunction
        Notification
        Transactions
        XmlListQuery
        GetCustomerCreditInformation
            docType
                GetCustomerCreditInformationRequestMessage
                GetCustomerCreditInformationResponseMessage
                schema_GetCustomerCreditInformationRequestMessage
                schema_GetCustomerCreditInformationRequestMessage_3
            process
PSFT_EnterpriseOne_AdapterServices
PSFT_EnterpriseOne_Interfaces
PSFT_JDBCAdapterServices
PSFT_OP_Adapter
PSFT_OP_Adapter_samples
```

Recent Elements:

- CompanyInfoInput
- AddChangeElectronicAddressPhone
- intToString
- dynamicServiceInvocation

process

PSFT_EnterpriseOne_AdapterSamples.GetCust...

SEQUENCE

Specification Reference

Input ationRequestMess Output nResponseMess

Validate input Validate output

Input	Output
payload	payload
auth	SOAP-FAULT
_port	

Properties: process

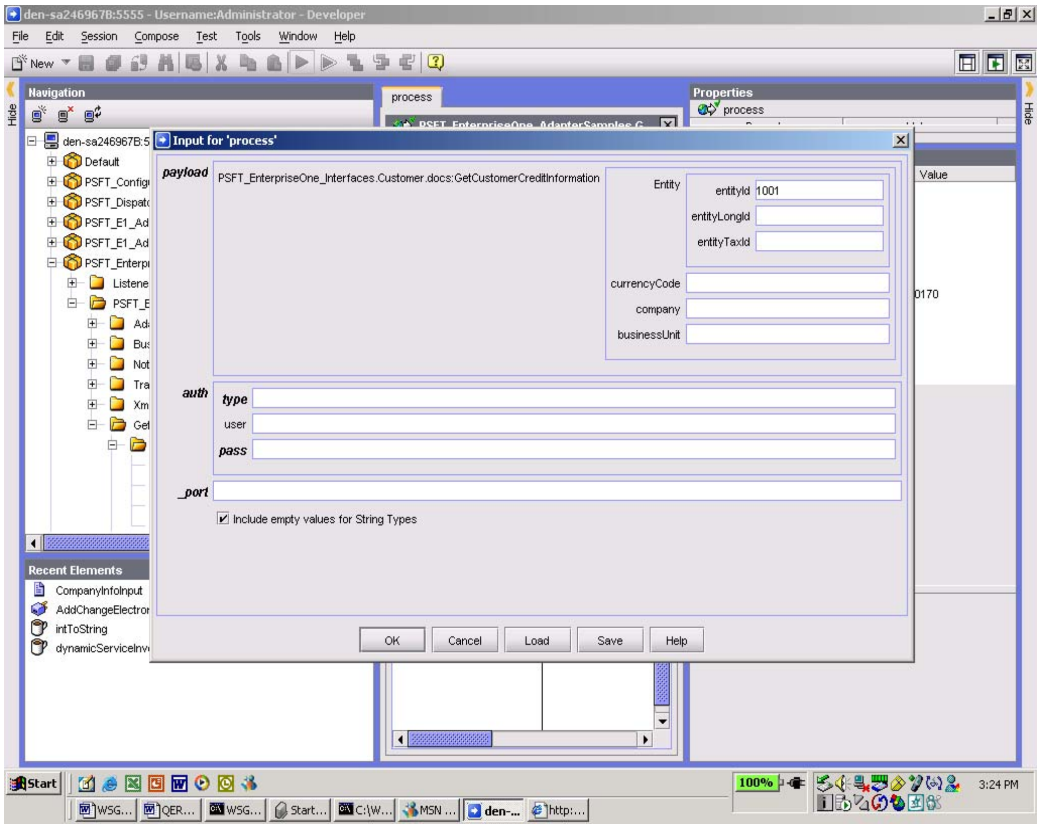
Property	Value
Runtime	
Stateless	False
Cache results	False
Cache expire	15
Reset cache	Reset
Prefetch	False
Prefetch acti	1
Execution loc	[$null] N...
Universal na	
Namespace r	

Results

Name	Value

Running the WSG Connector Service

Select the "process" flow you just created and click the run icon. Enter a valid entityID, check the "Include empty values for String Types, and click OK.


den-sa246967B:5555 - Username:Administrator - Developer
File Edit Session Compose Test Tools Window Help
Navigation
process
Properties
Input for 'process'
OK
Cancel
Load
Save
Help

payload PSFT_EnterpriseOne_Interfaces.Customer.docs:GetCustomerCreditInformation

Entity entityId 1001
entityLongId
entityTaxId
currencyCode
company
businessUnit

auth type
user
pass

_port

☑ Include empty values for String Types

CONCLUSION

Through the use of Service Oriented Architecture, Oracle provides companies the ability to find additional value out of their independent applications. Oracle's embracing of the concepts such as service discovery, consumption, orchestration and standards support, all of which were discussed throughout this paper, enable an enterprise-wide integration model that delivers a reusable suite of interoperable services in a standard's based environment.

- Products like Integration Repository can be used to discover over 2000+ Business Services that are implemented in the Oracle Applications.
- Oracle Applications are service enabled and can interoperate in a web service based environment through the use of standards such as SOAP and WSDL.
- With Oracle BPEL Process Manager, companies can orchestrate cross application business processes leveraging such standards as BPEL.

Oracle Applications and Technology are your Strategic assets that bridge the fundamental gaps between business requirements and IT capabilities. Companies can now realistically achieve their business goals by implementing a best-practice service-oriented IT architecture as the foundation of their future success based on Oracle Applications and Oracle Fusion Middleware.

APPENDIX A

```
package oracle.apps.fnd.wf.bes.sample;
import com.oracle.bpel.client.delivery.DeliveryService;
import java.util.Properties;
import java.util.Map;

//JNDI library for looking up for remote RMI server
import javax.naming.InitialContext;

//BPEL client code
import com.oracle.bpel.client.ClientDefs;
import com.oracle.bpel.client.Locator;
import com.oracle.bpel.client.NormalizedMessage;

//BES library
import oracle.apps.fnd.wf.bes.BusinessEvent;
import oracle.apps.fnd.wf.bes.BusinessEventException;
import oracle.apps.fnd.wf.bes.SubscriptionInterface;
import oracle.apps.fnd.wf.bes.server.Subscription;
import oracle.apps.fnd.wf.common.WorkflowContext;

/**
 * This class demostrates how to invoke a BPEL process from a BES
subscription
 */
public class InvokeBPELSub  implements SubscriptionInterface
{
  public void onBusinessEvent(Subscription eo, BusinessEvent event,
    WorkflowContext context) throws BusinessEventException
  {
    try{
        // properties in the classpath
        // You can also store these values as Profile options and
access them in the subscriptions
        // This way you can avoid hardcoding this information in
every subscription
        Properties props = new java.util.Properties();
        //sets target bpel platform, oracle installation, this will
be oc4j_10g
        props.setProperty("orabpel.platform","oc4j_10g");
        //sets jndi factory class, this is used to locate remote RMI
server

props.setProperty("java.naming.factory.initial","com.evermind.server.
rmi.RMIInitialContextFactory");
        //sets target url for target rmi server (oc4j
props.setProperty("java.naming.provider.url","ormi://127.0.0.1/orabpe
l");
        //sets user name for the remote server
     //For security reasons, you must store this information in your
own tables. Make sure you store them encrypted and decrypt the value
in the subscription
        props.setProperty("java.naming.security.principal","admin");
        //sets password for the remote server

props.setProperty("java.naming.security.credentials","welcome");

        System.out.println(props);

        Locator locator = new Locator("default", "bpel", props);
        DeliveryService deliveryService =
(DeliveryService)locator.lookupService(DeliveryService.SERVICE_NAME);
        //Object deliveryService =
locator.lookupService(DeliveryService.SERVICE_NAME);
        //
        //Maps event data into input message for BPEL process.
        //
        //Here you need to map you event parameters into BPEL input
message if any
        //ssn property is given by the Event definition
        //when an event is raisen, application can associate any
number of parameters
        //to it. As to what parameters are available for a given
event, it is upto to
        //the owner of the event.
        //
        //In this example, we assume social security number is one of
the event parameter,
        //which will be used as input for BPEL process
"CreditRatingService"
        //
        String ssn = event.getStringProperty("ssn");

        // construct the normalized message and send to bpel server
        String xml = "<ssn xmlns=\"http://services.otn.com\">" + ssn
+ "</ssn>";
        NormalizedMessage nm = new NormalizedMessage( );
        //payload is specific to the process you invoke, it is
defined in the WSDL used by your BPEL process

        nm.addPart("payload", xml );

        //Start the process
        // First parameter(CreditRatingService) is the name of 
portType as defined in the WSDL,
        // which is deployed to the BPEL Server
        // Second parameter (process) is the operation name that is 
associated to a receive activity
        // The third parameter (nm) is the context properties for the 
request.
        //
        //Depending on the process definition, if you expect a reply 
from the process, you should
        //use request of the deliveryService else, use post with the 
same arguments
        // e.g. NormalizedMessage res = 
deliveryService.post("CreditRatingService", "process", nm);
        //

        NormalizedMessage res = 
deliveryService.request("CreditRatingService", "process", nm);

        //Since we used request, we can retrieve the return message
        Map payload = res.getPayload();

        //you can store the result back from BPEL process into 
business event if it is required
        // string value of payload should be something like
        // <rating xmlns="http://services.otn.com">560</rating>
        event.setStringProperty("creditRate", ""+
payload.get("payload"));

        System.out.println("Credit Rate:"+payload.get("payload"));
    }catch(Exception e)
    {
      //
      //propagate the exception to BES, the last flag indicate this 
exception is
      //a true error so BES will invoke defined error handling logic
      //
      throw new BusinessEventException("Failed to start BPEL 
process", e, true);
    }
  }
}
```

APPENDIX B

```
package oracle.apps.fnd.wf.bes.sample;

import java.sql.Connection;
import java.util.Date;
import javax.sql.DataSource;
import oracle.apps.fnd.wf.bes.BusinessEvent;
import oracle.jdbc.pool.OracleDataSource;

/**
 * Sample code to programmatically raise an event
 */
public class RaiseEvent
{
 static String mEvent = "oracle.apps.fnd.wf.bes.sample.bpel.invoke";
  public static void main(String[] args)
   {
    Connection conn=null;
    try
    {
      conn=getJDBCConnection();
      //takes event name and event key
      BusinessEvent businessEvent =
        new BusinessEvent(mEvent, ""+System.currentTimeMillis());

      businessEvent.setStringProperty("ssn","123-23-2331");

      Date date= new Date();
      businessEvent.setSendDate(date);
      businessEvent.raise(conn);
    }
    catch (Exception e)
    {
      e.printStackTrace();
    }finally
    {
      if(conn!=null)
        try
        {
          conn.close();
        }
        catch (Exception e)
        {
        }
    }
  }
  private static Connection getJDBCConnection() throws Exception
  {
    OracleDataSource ods = new OracleDataSource();
    // Connection information to the Oracle E-Business Suite instance
    ods.setUser("apps");
    ods.setPassword("appspwd");
    ods.setURL("jdbc:oracle:thin:@host:port:sid");
    Connection sConn = ods.getConnection();
    sConn.setAutoCommit(false);
    return sConn;
  }
}
```

Oracle Applications Integration Cookbook: A Developer's Guide
August 2005
Author: Sharath Dorbala, Rajan Modi, Sebastian Artigas, Doron Avizov, Aali Masood, Meera Srinivasan

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com