ORACLE
PEOPLESOFT ENTERPRISE

Enterprise PeopleTools 8.50 PeopleBook: Integration Broker Administration

September 2009

ORACLE

Enterprise PeopleTools 8.50 PeopleBook: Integration Broker Administration
SKU pt850pbr0

Trademark Notice

License Restrictions Warranty/Consequential Damages Disclaimer

Warranty Disclaimer

Restricted Rights Notice

Hazardous Applications Notice

Third Party Content, Products, and Services Disclaimer

Contents

Chapter 4

Chapter 5

Chapter 7

Chapter 8

Chapter 9

Chapter 10

PeopleSoft Integration Broker Administration Preface

This preface provides an overview of the Peoplesoft Integration Broker Administration PeopleBook.

PeopleSoft Integration Broker Administration

This PeopleBook describes administration tasks for PeopleSoft Integration Broker.

This PeopleBook provides information for setting up and configuring integration system components, such as messaging servers, nodes, integration gateways, listening and target connectors, administer publication and subscription domains, and so on, and describes how to enable the integration system for handling services.

It also provides information on taking measures to secure the integration environment by applying security on the main integration system components.

This PeopleBook also describes methods to enhance and fine-tune integration system performance.

PeopleBooks and the Online PeopleSoft Library

A companion PeopleBook called PeopleBooks and the Online PeopleSoft Library contains general information, including:

- Understanding the PeopleSoft online library and related documentation.
- How to send PeopleSoft documentation comments and suggestions to Oracle.
- How to access hosted PeopleBooks, downloadable HTML PeopleBooks, and downloadable PDF PeopleBooks as well as documentation updates.
- Understanding PeopleBook structure.
- Typographical conventions and visual cues used in PeopleBooks.
- ISO country codes and currency codes.
- PeopleBooks that are common across multiple applications.
- Common elements used in PeopleBooks.
- Navigating the PeopleBooks interface and searching the PeopleSoft online library.
- Displaying and printing screen shots and graphics in PeopleBooks.
- How to manage the PeopleSoft online library including full-text searching and configuring a reverse proxy server.
- Understanding documentation integration and how to integrate customized documentation into the library.

- Glossary of useful PeopleSoft terms that are used in PeopleBooks.

You can find this companion PeopleBook in your PeopleSoft online library.

Chapter 1

Getting Started with PeopleSoft Integration Broker Administration

This chapter provides an overview of PeopleSoft Integration Broker Administration and discusses considerations for how to:

- Plan the integration architecture.
- Understand integrations processed by the integration system.
- Determine security.
- Plan for support.
- Assess staff skills.

PeopleSoft Integration Broker Administration Overview

This PeopleBook describes how to perform system administration tasks in PeopleSoft Integration Broker such as:

- Set up and configure integration system components, such as messaging servers, nodes, integration gateways, listening and target connectors, and so on.
- Configure the integration system to handle services, including specifying namespaces, setting up UDDI repositories, and so on.
- Secure the integration environment by applying security at the web server, gateway, application server, node and service operation level.
- Fine tune integration system performance by employing failover, master/slave processing, load balancing, and so on.
- And more.

Administering Peoplesoft Integration Broker

PeopleSoft Integration Broker is installed as part of the PeopleTools installation process. Information about configuring the integration gateway, creating service operations and administering integrations is described later in this PeopleBook. This section provides information to consider before you begin to use PeopleSoft Integration Broker.

Planning the Integration Architecture

The two major components of PeopleSoft Integration Broker are the integration gateway and the integration engine. The integration gateway is a platform that manages the receipt and delivery of messages passed among systems through PeopleSoft Integration Broker. The integration engine is an application server process that routes messages to and from PeopleSoft applications as well as transform the structure of messages and translates data according to specifications that you define.

Evaluate historical integration data, current data, as well as expected growth and increased traffic. Consider how many interfaces you have in production and how much system resources they use. Also consider how many of these interfaces will remain nightly batch file loads versus how many do you want to be real-time service based integrations. Devise simulated real-life integration scenarios where you can estimate volume and size of the transactions to a certain degree. Then use this information for benchmarking and stress testing, which should lead to performance tuning, hardware sizing, and so on.

Understanding Integrations Processed by the Integration System

Work with development teams to understand the type, number and frequency of integration that will be processed on the system. Doing so will assist you in setting up and configuring components properly, as well as in performance tuning the system.

Consider the following:

- Real-time integrations or scheduled integrations.
- Determine if your business needs require using real-time integration, scheduled integrations, or a combination of both. Scheduled batch processing and file loads of scheduled integration may impact system performance and the running of other system applications.
- Inventory the integration being developed and performed.
- Determine which systems and applications will participate in each integration. Consider dependencies on other systems owned by other groups having concurrent releases, and data dependencies within the context of synchronizing data between systems. Do you need permission from business owners to integrate with their systems?
- Synchronous integrations and asynchronous integrations.

  In PeopleSoft Integration Broker synchronous integrations, all processing stops after the system sends a request to an integration partner, until a response is received back from that partner. In PeopleSoft Integration Broker asynchronous integrations, each request is placed in a queue to be processed as soon as the system can accommodate the request.

  Synchronous integration processing and asynchronous integration processing each place different loads on the integration system. Understanding the processing that takes place on your system can help you better tune the system for optimal performance.

Determining Security

Unlike a public web service on the internet that retrieves a stock quote for a given ticker symbol, the web services and integrations in your PeopleSoft applications can expose sensitive information such as financial data. PeopleSoft Integration Broker facilitates transfer of information between systems; however, a security analyst must evaluate security requirements for each individual integration.

For example, security requirements might differ when interfacing with credit card processing vendors, versus publishing salary information out of human resources, versus synchronizing business units between applications, and so on. Perhaps certain information should be available to the public, including systems outside of your company, such as how many inventory items are available for sale. Other information might be restricted to internal employees only, internal application systems only, or perhaps only certain users of a particular application system.

PeopleSoft Integration Broker allows you to secure each individual integration to the level of security required as well as all integration data flowing over the wire.

Accessing Staff Skills

Administrators of PeopleSoft Integration Broker should have familiarity, training or experience in the following areas:

- PeopleTools.
- Web server administration.
- Application server administration.
- Performance testing and tuning knowledge.

Chapter 2

Understanding Setting Up PeopleSoft Integration Broker

This chapter provides the high-level steps to set up PeopleSoft Integration Broker. This chapter discusses how to:

- Determine the messaging architecture.
- Install PeopleSoft Integration Broker.
- Install web servers.
- Install application databases.
- Install PeopleTools.
- Install the PeopleSoft Pure Internet Architecture.
- Configure and start messaging servers for asynchronous messaging.
- Activate pub/sub server domains.
- Define integration gateways and load connectors.
- Configure integration gateway properties.
- Configure the integration system to handle services.

See Also

Chapter 1, "Getting Started with PeopleSoft Integration Broker Administration," page 1

Enterprise PeopleTools 8.50 PeopleBook: PeopleSoft Integration Broker, "Understanding PeopleSoft Integration Broker"

Determining the Messaging Architecture

A key step in creating and implementing integrations is to determine what systems to integrate and the architecture to use. For example, your purpose might be to integrate with other PeopleTools 8.50 systems where a firewall is involved, integrate with third-party systems, or integrate with PeopleSoft 8.1x systems.

The *Enterprise PeopleTools 8.50 PeopleBook: Integration Broker* features an appendix that provides overview information about several messaging architecture scenarios.

See Also

Enterprise PeopleTools 8.50 PeopleBook: PeopleSoft Integration Broker, "Integration Scenarios"

Installing PeopleSoft Integration Broker

PeopleSoft Integration Broker components are installed during the PeopleTools installation and the PeopleSoft Pure Internet Architecture installation.

The PeopleSoft Integration Broker integration engine is installed during the PeopleTools installation process.

See Chapter 2, "Understanding Setting Up PeopleSoft Integration Broker," Installing PeopleTools, page 7.

The integration gateway is installed as part of the PeopleSoft Pure Internet Architecture installation process.

See Chapter 2, "Understanding Setting Up PeopleSoft Integration Broker," Installing the PeopleSoft Pure Internet Architecture, page 7.

Installing Web Servers

To install and run PeopleTools, you must install a web server.

See Also

PeopleTools 8.50 Install Guide for your database

Your web server documentation.

Installing Application Databases

After you install PeopleTools, install your application database.

See Also

PeopleTools 8.50 Install Guide for your database.

Installing PeopleTools

PeopleSoft Integration Broker is installed as part of the PeopleTools installation process. The PeopleTools installation process also installs the executable file you need to install the PeopleSoft Pure Internet Architecture.

See Also

PeopleTools 8.50 Install Guide for your database.

Installing the PeopleSoft Pure Internet Architecture

Run the PeopleSoft Pure Internet Architecture setup program. The executable file is provided as part of the PeopleTools installation.

Before attempting to start the PeopleSoft Pure Internet Architecture, verify that the web server is running; the web server must be running to start the PeopleSoft Pure Internet Architecture.

See Also

PeopleTools 8.50 Install Guide for your database.

Configuring and Starting Messaging Servers for Asynchronous Messaging

Before using PeopleSoft Integration Broker for asynchronous integrations, you must configure and start the messaging server using PSADMIN.

See Chapter 4, "Administering Messaging Servers for Asynchronous Messaging," page 15.

Activating Pub/Sub Server Domains

You must activate the domain on which the pub/sub server resides before you can use the messaging server.

To activate pub/sub server domains, use the Quick Configuration page . In the Integration Broker Domains section of the page, locate your machine name and select Active from the drop-down list box and click the Save button.

See Also

Chapter 3, "Using the Integration Broker Quick Configuration Page," page 11

Chapter 10, "Managing Pub/Sub Server Domains," page 151

Defining Integration Gateways and Loading Connectors

PeopleSoft Integration Broker is delivered with one local gateway, *LOCAL,* defined. You can use this gateway as the default local gateway, or create a new gateway and designate that one as the default local gateway.

After you access the delivered local gateway or create your own, you must specify its URL and save the changes. The gateway URL is typically the following:

```
http://<machine_name>:<port>/PSIGW/PeopleSoftListeningConnector
```

The integration gateway URL is case sensitive.

Next you must click the Load Gateway Connectors button to load the connectors delivered with PeopleSoft Integration Broker.

See Also

Chapter 3, "Using the Integration Broker Quick Configuration Page," page 11

Chapter 5, "Managing Integration Gateways," page 29

Configuring Integration Gateway Properties

After you define the default local integration gateway, specify the integration gateway URL and load the delivered connectors, there are additional required and optional gateway properties to set. You set these properties using the integrationGateway.properties file.

Use one of the following methods to access the file:

- On the Quick Configuration page click the Advanced Gateway Setup link located under the Gateway URL field.
- On the Gateways page click the Gateway Setup Properties link located next to the integration gateway URL field.

At a minimum you must set the following in the integrationGateway.properties file:

- Set the Oracle Jolt connection string parameters in the DELIVERED CONNECTOR CONFIGURATION Section of the file. In most situations, you set the parameters under "JOLT connect string settings for Application Server(s) with known NODENAMEs."

- Specify and encrypt the keystore password.

See Also

Chapter 3, "Using the Integration Broker Quick Configuration Page," page 11

Chapter 5, "Managing Integration Gateways," Accessing the integrationGateway.properties File, page 42

Chapter 5, "Managing Integration Gateways," Configuring Security and General Properties, page 46

Configuring the Integration System to Handle Services

To create services, service operations and generate WSDL documents, you must configure the system to handle services.

PeopleSoft Integration Broker features a Services Configuration page where you must specify the following items before you can create and work with services: services namespace, schema namespace and target location.

See Also

Managing Services, Configuring PeopleSoft Integration Broker for Handling Services

Chapter 3

Using the Integration Broker Quick Configuration Page

This chapter discusses using the Integration Broker Quick Configuration page to set up and access PeopleSoft Integration Broker configuration properties.

Prerequisites for Using the Integration Broker Quick Configuration Page

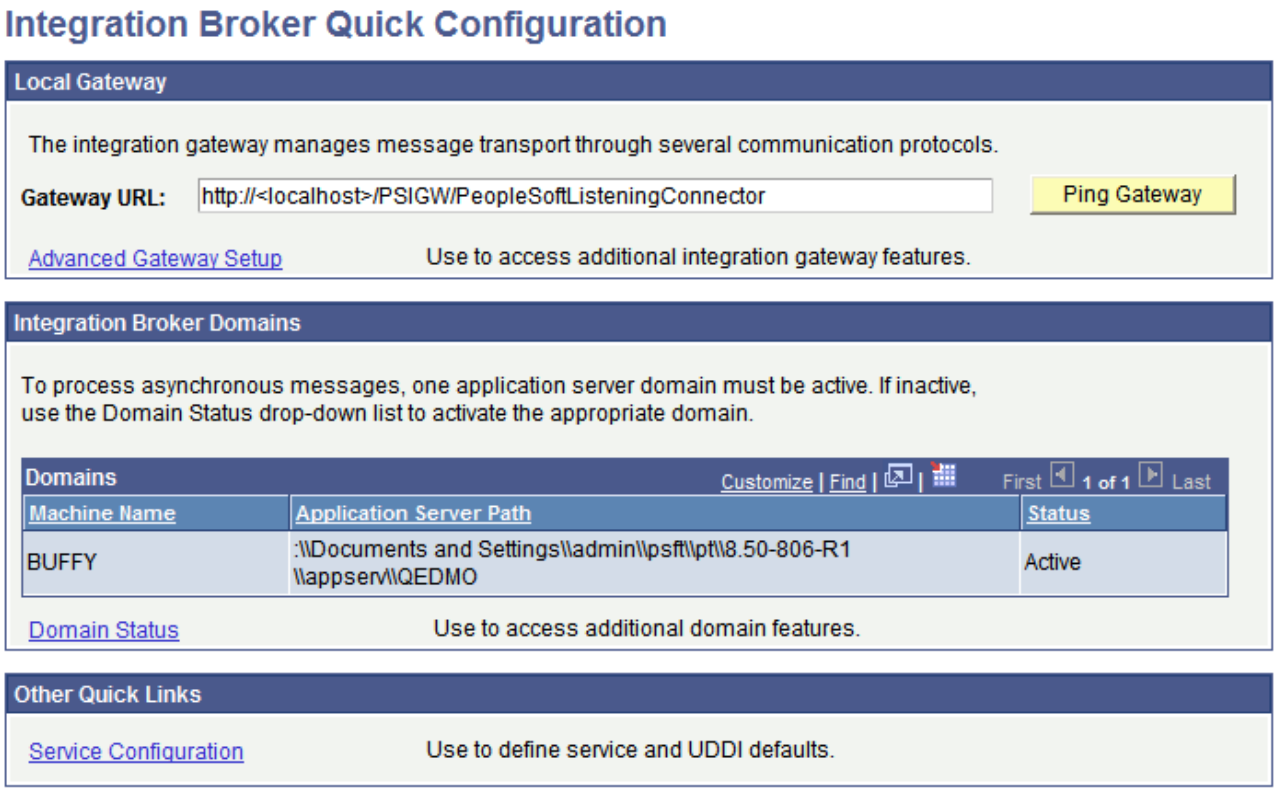
Before you perform the tasks described in this chapter, install PeopleTools, and configure and start the application server. In addition, install, configure, and start the web server.

See Also

PeopleTools Installation Guide for your database

Accessing the Integration Broker Quick Configuration Page

You can set up and access most PeopleSoft Integration Broker configuration properties using the Integration Broker Quick Configuration page (PTIB_ADMIN). To access the page, select PeopleTools, Integration Broker, Configuration, Quick Configuration.



Integration Broker Quick Configuration page

The page provides access to the following configuration properties.

Gateway URL

Enter the integration gateway URL in the following form:

```
http://<machinename>:<port>/PSIGW/PeopleSoftListening⇒
Connector
```

By default the port number is *80* for HTTP and *443* for HTTPS. If using the default port number, you do not need to specify it in the URL.

For HTTPS, the URL should start with *https.*

The integration gateway URL is case-sensitive.

Ping Gateway

Click the button to verify that the integration gateway is responding. If active, a window appears that displays the name of the active target connector, the PeopleTools version you are running, and the status of *Active.*

Advanced Gateway Setup

Click the link to access the Gateways page where you load target connectors and specify their properties. Use this page to also specify nodes with which the gateway will communicate and access the integrationGateway.properties file to set additional properties.

See Chapter 5, "Managing Integration Gateways," Setting General Connection Properties, page 47.

(Domain) Status Click Active from the drop-down list box to activate pub/sub servers on an application server domain.

You must activate the pub/sub servers on application server domains used for messaging before you can use them to successfully send and receive messages.

The drop-down list box appears only for domains that are currently inactive.

Domain Status Click the link to access the Domain Status page in the Service Operations Monitor where you can set domain grace periods, set domain failover, view dispatcher status, and more.

See Chapter 10, "Managing Pub/Sub Server Domains," page 151.

Service Configuration Click the link to access the Service Configuration page to set required services properties, such as service namespace and schema namespace.

This link also provides access to set required properties when using Universal Description, Discovery and Integration (UDDI) repositories to provide and consume web services.

See Chapter 8, "Configuring PeopleSoft Integration Broker for Handling Services," page 143.

See Also

Enterprise PeopleTools 8.50 PeopleBook: PeopleSoft Integration Broker, "Understanding PeopleSoft Integration Broker Metadata"

Chapter 4

Administering Messaging Servers for Asynchronous Messaging

This chapter provides an overview of messaging server administration and discusses how to:

- Create and assign dedicated servers.
- Edit messaging server queue lists.
- Delete messaging servers.
- Configure messaging servers.
- Set the Oracle Tuxedo queue size.

Understanding Messaging Server Administration

This section discusses messaging servers, messaging server processes, and dedicated messaging servers.

Messaging Servers

The PeopleSoft messaging infrastructure is the core system upon which PeopleSoft Integration Broker is built. Before using Integration Broker for asynchronous message processing, you must configure and start the messaging server.

Note. The messaging servers and messaging server processes are used for asynchronous integrations only. If you are performing only synchronous integrations, you need not configure a messaging server.

Activating Messaging Server Domains

Pub/sub server domains are delivered inactive, and you must activate them for the pub/sub system to become available.

You can activate pub/sub server domains using the Integration Broker Quick Configuration page or on the Domain Status page in the Service Operations Monitor.

See Chapter 3, "Using the Integration Broker Quick Configuration Page," page 11.

See Chapter 10, "Managing Pub/Sub Server Domains," page 151.

Messaging Servers in the DB2 UDB OS/390 and z/OS Environments

For DB2 UDB OS/390 and z/OS environments, PeopleSoft delivers messaging servers with persistent cursors off. Therefore, all SQL statements are compiled each time they are invoked.

To change the persistent cursors setting:

1. In PSADMIN locate the Values for config section — Publish&Subscribe.
2. Set the Persistent Cursors on DB2/OS390 option. The values are:
   - *0*: Persistent cursors off.
   - *1*: Persistent cursors on.

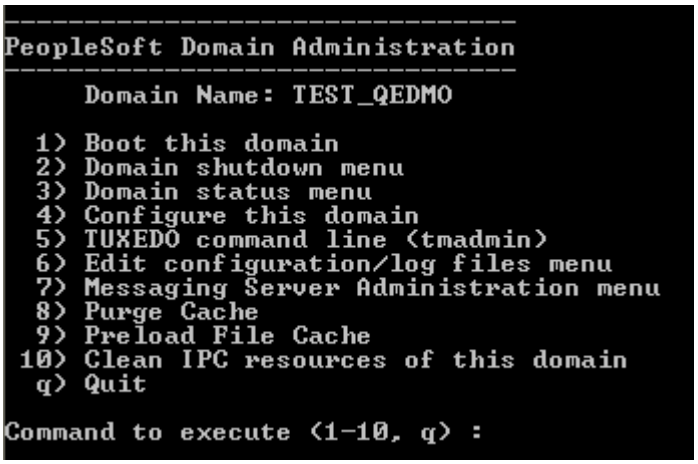
Messaging Server Processes

Although the server processes devoted to the messaging system are all part of the larger application server domain, they comprise a distinct set of processes that aren't involved with the ordinary transactions associated with PeopleSoft Pure Internet Architecture connections.

Six processes of two types—dispatchers and handlers—are paired to produce the messaging servers that transmit asynchronous messages throughout the messaging system. A set of three messaging servers—a publication broker, a publication contractor, and a subscription contractor—is required by PeopleSoft Integration Broker. The following table lists the generic names for the processes:

Messaging Server	***Dispatcher Name***	***Handler Name***
Publication Broker (BRK)	PSBRKDSP	PSBRKHND
Publication Contractor (PUB)	PSPUBDSP	PSPUBHND
Subscription Contractor (SUB)	PSSUBDSP	PSSUBHND

To distinguish the messaging servers, the PeopleSoft Server Administration utility (PSADMIN) includes a separate menu for administering them—the Messaging Server Administration menu. You select this menu from the PeopleSoft Domain Administration menu, as shown in the following example:



PeopleSoft Domain Administration menu

From this menu, you can create new messaging servers, edit the queue list for existing messaging servers, and delete messaging servers that are no longer needed.

Note. Although you add new messaging servers using a separate menu, you configure the messaging server processes with PSADMIN as you would any other server process.

See Also

Enterprise PeopleTools 8.50 PeopleBook: System and Server Administration, "Using PSADMIN Menus"

Understanding Dedicated Messaging Servers

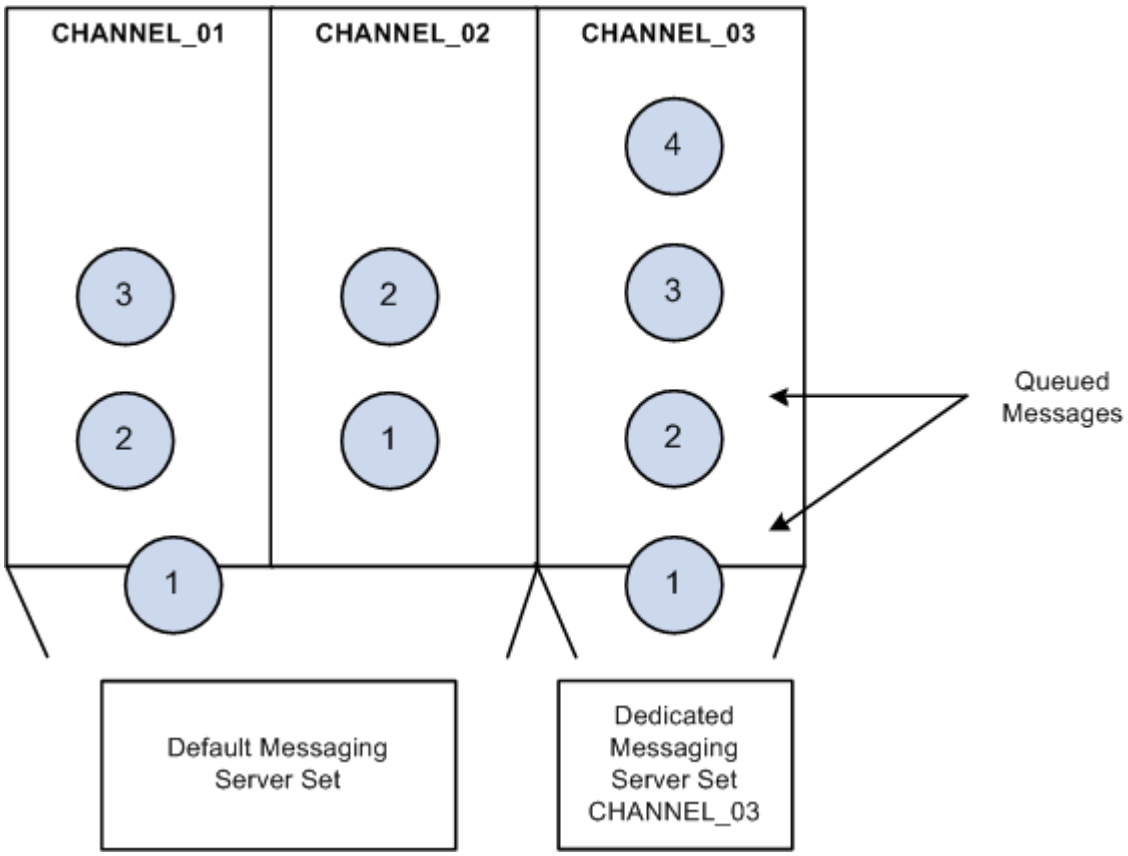
When you create a new application server domain, PSADMIN offers a set of messaging server processes that comprise the default messaging server set for that domain. The default messaging server set is sufficient for development, testing, or demonstrations.

You might use the default messaging server set as the only messaging server set; however, in most cases, it is insufficient. As the volume of published messages increases in a production system, it's likely that a single messaging server set will become overloaded. To avoid potential overloads and performance degradation, create additional dedicated messaging servers to cope with an increase in message volume.

Note. Dedicated messaging servers are used only for asynchronous messaging.

When you create a new messaging server, you assign it to a particular queue using PSADMIN. If a given queue is the most active and creates performance bottlenecks, you can dedicate several messaging servers to that queue to cope with the message volume. A messaging server is capable of handling multiple message queues.

The following illustration depicts a dedicated messaging server set assigned to QUEUE_03:



Dedicated messaging server set

In this scenario, the default messaging server set (_dflt process collection) continues to process the messages in the other message queues while the dedicated messaging server set processes only the messages within a specified queue. Unless you create and configure dedicated messaging servers, the default messaging server set handles all incoming messages. Remember that a messaging server set is a collection of six messaging server processes.

Note. Before you can assign messaging servers to message queues, you must first define the message queues using PeopleSoft Application Designer.

The process for adding a dedicated messaging server includes two parts:

- Creating the new messaging server.

  Use the Messaging Server Administration menu in PSADMIN. This is where you specify the type of server you're adding, name the server, and assign it to specific message queues.

- Configuring the new messaging server.

  When you add a new messaging server of any type, the configuration files are updated to include parameters for the new server processes. Because a messaging server consists of two server processes, when you create a new one, you'll see two additional configuration sections in the PSADMIN domain configuration menu. They appear identical to the _dflt messaging server processes, except they have the name that you gave them in place of the *_dflt*. For any new messaging server processes to take effect, you must first reconfigure the domain to include the new parameters.

Note. Typically, you add multiple messaging elements simultaneously, so you should create all the elements and then reconfigure the domain once.

Considerations When Creating Dedicated Servers

When creating dedicated messaging servers, consider the following points:

- There is no validation checking when you enter service operation queue names in PSADMIN. As a result, if service operation queue names are not spelled correctly and match those defined in the system, the dedicated server will not process any service operation. Instead the default server will process them.
- Never split a service operation queue across domains. You don't want a situation where a service operation queue is assigned to Domain A and the same service operation queue is also assigned to Domain B, since both domains will try to do the same work. You want specific service operation queues for Domain A and specific service operation queues for Domain B.
- Setting up a dedicated server consists of a creating a dedicated dispatcher and handler(s). Make sure that the number of handlers booted is sufficient to process the request volume.
- If you create more than one dedicated server over different domains do not to include any service operation queues already specified for other dedicated servers of the same server type. For example, do not include Service Operation Queue A in Publication Broker Server X, as well as in Publication Broker Server Y.
- Verify that the Oracle tuxedo queue size is large enough and correctly configured in PSADMIN.

  See Chapter 4, "Administering Messaging Servers for Asynchronous Messaging," Setting the Oracle Tuxedo Queue Size, page 27.
- If you choose to set up group domain failover for dedicated servers, ensure that:
  - Service operation queue sets within groups are identical.
  - Service operation queue sets between groups are unique.

    See Chapter 12, "Tuning Messaging System Performance," Setting Up Domain Failover, page 255.
- When you create a messaging server, the following dispatcher parameters are populated with their default values. Verify those default settings you want to keep and those that you want to change.
  - Restart period.
  - Scan interval.
  - Dispatcher queue maximum queue size.
  - Memory queue refresh rate.

    See Chapter 4, "Administering Messaging Servers for Asynchronous Messaging," Specifying Dispatcher Parameters, page 23.

Creating and Assigning Dedicated Servers

Typically, you create one server of each type to produce a complete messaging server set dedicated to one or more service operation queues.

Note. Although a messaging server set consists of one of each of the three server types, they do not all need to be dedicated servers. For example, for a given service operation queue, you can create only a dedicated publication contractor. If you haven't assigned a dedicated publication broker or a dedicated subscription contractor to the service operation queue, the default publication broker and subscription contractor is used.

The following example shows the Message Server Administration menu:

```
------------------------------------------------------------
            Messaging Server Administration menu
------------------------------------------------------------
  Domain Name : QEDMODW

  In addition to the default messaging servers, the following
  dedicated messaging servers are in the domain configuration:

       << No dedicated messaging servers are defined >>


  Commands:
   1) Create a new messaging server
   2) Edit the channel list for a messaging server
   3) Delete an existing messaging server
   q) Quit

Command to execute (1-3, q) : 1
  1) Publication broker
  2) Publication contractor
  3) Subscription contractor

Enter the number of the type you want to create ('q' to cancel):
```

Creating a new messaging server

To create a dedicated messaging server:

1. From the PeopleSoft Domain Administration menu, select the Messaging Server Administration menu.
2. From the Messaging Server Administration menu, select the *Create a new messaging server.*
3. From the submenu that appears, select the type of server to create.

   You can create a publication broker, a publication contractor, or a subscription contractor.
4. Enter a name to identify the new messaging server.

   The name is limited to six characters; for example, *PT8MSG*. The name that you enter is appended to each generic server process name; for example, PSBRKDSP_PT8MSG for the broker dispatcher and PSBRKHND_PT8MSG for the broker handler.

   Note. The name that you enter must be unique for the messaging server type in the current domain.

5. Specify the service operation queue that is handled by the new messaging server.

   You must specify a service operation queue, which must already be defined in the PeopleSoft Pure Internet Architecture.

   Note. The service operation queue name that you enter must exactly match the name that appears in the PeopleSoft Pure Internet Architecture. No prompt or validation occurs between PSADMIN and PeopleSoft Pure Internet Architecture definitions.

   Important! Don't specify a given service operation queue for more than one messaging server of each type in the current domain. For example, you cannot have two subscription contractors assigned to the service operation queue. Nor can you have two dispatchers assigned to the service operation queue.

After several status messages, the Messaging Server Administration menu reappears, displaying a list of the existing dedicated messaging servers for the current domain.

Editing Messaging Server Queue Lists

After you create a publication broker, publication contractor, or subscription contractor, you may need to add more service operation queues to the server's queue list, or you may want to decrease the number of service operation queues it services to improve performance. You use the commands shown in the following example:

```
------------------------------------------------------------
             Messaging Server Administration menu
------------------------------------------------------------
  Domain Name : PT848805I1

  In addition to the default messaging servers, the following
  dedicated messaging servers are in the domain configuration:


   SERVER NAME       TYPE  QUEUES
   ----------------  ----  ----------------
   TEST01            BRK   QE_FLIGHTPLAN_QUEUE
   TEST02            PUB   QE_PO_QUEUE


  Commands:
   1) Create a new messaging server
   2) Edit the queue list for a messaging server
   3) Delete an existing messaging server
   q) Quit

Command to execute (1-3, q) : 2


       SERVER NAME       TYPE  QUEUES
       ----------------  ----  ----------------
   1)  TEST01            BRK   QE_FLIGHTPLAN_QUEUE
   2)  TEST02            PUB   QE_PO_QUEUE

Enter the number of the server to be edited: 2

Changing queue list for server 'TEST02'...

FORMAT: Alphanumeric, max 30-char queue names separated by commas (no tabs or spaces)
Current queues: [QE_PO_QUEUE]
Enter new queues:
```

Modifying a messaging server queue list

To modify a queue list:

1. From the PeopleSoft Domain Administration menu, select Messaging Server Administration menu.
2. From the Messaging Server Administration menu, select Edit the queue list for a messaging server.
3. From the list of defined servers, select the messaging server for which you want to modify the queue list.
4. Specify a list of the message queues that will be handled by the selected server.

   You must specify at least one message queue. Multiple queue names must be entered as a list separated by commas, with no spaces; for example, HRMS_01,HRMS_02,CRM_03.

   Note. The new list of message queues that you enter replaces the current list of queues for the selected messaging server. The queues that you specify must already be defined in the PeopleSoft Pure Internet Architecture.

After several status messages, the Messaging Server Administration menu reappears, displaying the updated messaging server listing.

Deleting Messaging Servers

Sometimes a previously created messaging server is no longer needed. Rather than allow the server to consume valuable system resources, you should remove it from the domain.

To delete a messaging server from a domain:

1. From the PeopleSoft Domain Administration menu, select Messaging Server Administration menu.
2. From the Messaging Server Administration menu, select Delete an existing messaging server.
3. From the list of defined servers, select the messaging server to delete.

   After several status messages, the Messaging Server Administration menu reappears, displaying the remaining dedicated servers.

Configuring Messaging Servers

Once you create dedicated messaging servers, you must configure their dispatcher and handler processes so that they boot when you start the application server. You configure these processes using PSADMIN, as you do other server processes that run on the application server. Before you configure additional messaging server processes, familiarize yourself with the other server processes that run on the application server.

See *Enterprise PeopleTools 8.50 PeopleBook: System and Server Administration*, "Using PSADMIN Menus."

Two types of server processes comprise each messaging server: a dispatcher and a handler. Each process type requires that you set a different set of parameters. Most of the parameters are similar to other server processes, such as PSAPPSRV, but some parameters are specific to messaging servers.

Note. The following sections also apply to the _dflt messaging server processes. Only one parameter is different for a dedicated messaging server process and its _dflt counterpart—the Queues parameter. That parameter enables you to add message queues to the queue list. The _dflt server processes cannot be associated with a specific message queue.

Specifying Dispatcher Parameters

There are three generic process types that are the basis for all dispatcher processes:

- PSBRKDSP, which is the publication broker dispatcher.
- PSPUBDSP, which is the publication contractor dispatcher.
- PSSUBDSP, which is the subscription contractor dispatcher.

The following parameters apply to all three process types.

Recycle Count

Specifies the number of times each dispatcher process is executed before being terminated (intentionally) by the system and then immediately restarted.

You should intermittently recycled servers to clear buffer areas. The time required to recycle a server is negligible (a matter of milliseconds), however part of the server initialization process is to rebuild the in-memory queues. The time to complete this is dependent on the number of messages residing in the pub/sub database table.

The Recycle Count parameter does not translate into a native Oracle Tuxedo parameter in the PSAPPSRV.UBB file. Instead, the value is stored in memory and is managed by the system.

Allowed Consec Service Failures (Allowed consecutive service failures)

This option enables dynamic server process restarts in the event of service failures.

To set this option, enter a number greater than *0*. To disable it, enter *0*. The default value for this parameter is *2*. The value that you enter is the number of consecutive service failures that cause a recycle of the server process. This is a catchall error handling routine that allows a dispatcher to terminate itself if it receives multiple, consecutive, fatal error messages from service routines. Such errors should not occur consecutively; however, if they do, it indicates that the server process needs to be recycled or cleansed. A retry message appears when the specified number of service failures occurs.

Dispatch List Multiplier

Limits the number of dispatched messages by the number you specify, multiplied by the number of associated handler(s). This parameter is useful for unordered queues when all messages could go out at once. The default value is *10*.

Scan Interval

Specifies the number of seconds between scans of the work queue when idle.

The default value is *15* seconds.

The scan interval is necessary to detect the following types of messages:

- Messages published from an application server domain that is not the active pub/sub domain as selected on the Domain Status page in the Service Operations Monitor.
- Cases where the broker server does not receive a notice of the publication.

When a message is in the queue, the broker server doesn't receive a notice of the publication. A scan interval is required to make sure these types of messages are processed in a timely manner. The scan interval is analogous to the polling that PeopleSoft Process Scheduler performs on the Process Request table. In addition, the scan interval detects messages that have been resubmitted—for example, after an error. Decreasing the scan interval decreases latency for these types of publishes and error recovery.

Note. The scan interval and ping rate (as a percentage) determines the actual interval for pinging any unavailable remote nodes. The algorithm used is: (attempts) x (ping rate) x (scan interval).

Ping Rate

Determines the number of seconds of inactivity before the server scans the database queues to restart any stalled or crashed items.

The default value is *150* seconds.

The ping rate is used in conjunction with the scan interval for pinging remote nodes. See the definition for Scan Interval in this section.

Maximum Ping Interval

Determines the maximum interval, in hours, between subsequent attempted pings of any unavailable remote nodes.

Dispatcher Queue Max Queue Size

Determines the maximum number of items per service operation queue that the dispatcher keeps in memory. The default value is *1000*.

Memory Queue Refresh Rate

PeopleSoft Integration Broker maintains current asynchronous messaging queues in system memory for quick access. Occasionally, these cached queues can become corrupted. At that point, they must be refreshed from the PeopleSoft Integration Broker data tables. The likelihood and frequency of cache corruption depends on a combination of factors specific to the messaging system. If you need to periodically refresh the in-memory queues, you can use this parameter to tailor the frequency of the refresh to fit the situation.

Each dispatcher on the system has its own queue. For each queue, you set the rate equal to the number of dispatch attempts that must occur before the queue is refreshed. The refresh occurs only when the specified number of dispatch attempts is reached for a given message queue.

For example, with a memory queue refresh rate of *8*, multiple queues could have up to seven dispatch attempts each without triggering any refresh. The following settings are also significant:

- A setting of *0* (the default) disables the refresh altogether.
- A setting of *1* triggers a refresh immediately after every dispatch attempt, effectively disabling memory caching.

Restart Period

Specifies the number of seconds between restart attempts on *Started* items in the work queue.

An item which stays in *Started* state for more than a few seconds might be stalled—for example, the service request might have been lost, or the handler might have crashed. Decreasing the restart period reduces the latency for recovering stalled items with the status *Started.* However, under high load, items might stay in the *Started* state longer than normal for valid reasons. All handlers might be busy, and the handler service request for the item might be queued at the Oracle Tuxedo level. Setting the restart period too low results in redundant restarts. The dispatcher dispatches the item again, even though the original request is still in the Tuxedo queue. A small number of extra restarts is benign; however, at higher volumes, the unnecessary restarts can fill up the queue and block real requests. The formula for a reasonable value for the restart period is:

((incoming requests per second) / (number of handlers)) × (average processing time per request)

For example, if you have an incoming rate of 20 per second, and you have four handlers, each handler is busy processing one item and will have four others waiting in the queue. A new item must wait for the currently processing item—plus the four items in the queue—before it is processed. If each item takes 10 seconds to process, the new item will stay in *Started* status for approximately 50 seconds before the handler works on it. If it stays in *Started* status longer, it's likely that the request to the handler has been lost, and the item should be restarted.

Note. Using a value greater than *3540* for the dispatcher restart period results in constant restarts.

Specifying Messaging Server Process Handler Parameters

There are three generic process types that are the basis for all handler processes:

- PSBRKHND, which is the publication broker handler.
- PSPUBHND, which is the publication contractor handler.
- PSSUBHND, which is the subscription contractor handler.

The following parameters apply to all three process types.

Min Instances (Minimum instances)	Specifies the number of handler server processes started at boot time.
Max Instances (Maximum instances)	Specifies the maximum number of handler server processes that can be started or spawned.
Service Timeout	Specifies the number of seconds a handlers waits for a service request before timing out. Service timeouts are recorded in the TUXLOG and APPSRV.LOG. In the event of a timeout, the handler terminates itself and Oracle Tuxedo automatically restarts the process.
Recycle Count	Specifies the number of times that the system executes each server before the PeopleSoft system intentionally terminates the process. Server processes must be intermittently recycled to clear buffer areas. The time required to recycle a server is negligible (a matter of milliseconds). The Recycle Count parameter does not translate into a native Oracle Tuxedo parameter in the PSAPPSRV.UBB file. Instead the value is stored in memory and is managed by the PeopleSoft system.
Allowed Consec Service Failures (Allowed consecutive service failures)	This option enables dynamic server process restarts in the event of service failures. To set this option, enter a number greater than 0. To disable it, enter *0.* The default for this parameter is *2.* The numerical value that you enter is the number of consecutive service failures that cause a recycle of the server process. This is a catchall error handling routine that allows a handler to terminate itself if it receives multiple, consecutive, fatal error messages from service routines. Such errors should not occur consecutively; however, if they do, it indicates that the server process needs to be recycled or cleansed. A retry message appears when the specified number of service failures occurs.
Max Retries (Maximum retries)	Specifies the maximum number of times that the server attempts to restart a failed action. This parameter prevents a bad item from continuously crashing a handler process. The counter is incremented when the handler sets the status to *Working* but before it actually starts processing the item.

Setting the Oracle Tuxedo Queue Size

The messaging system uses the Tuxedo queue size indicated in the application server domain section of PSADMIN to determine when the Tuxedo queue size has reached its maximum. The pub/sub system reads the actual queue size periodically, based on the Tuxedo Queue Status Check Count parameter. The system throttles itself so that it does not exceed this maximum, thereby preventing queue saturation and degraded performance.

Set the Tuxedo Queue Size parameter equal to that of the kernel parameter used by the machine running the pub/sub processes (msgsys:msgingo_msgmax).

To set the Tuxedo queue size for the messaging system:

1. In PSADMIN navigate to the Values for config section – PSAPPSRV part of the file. To do so:
   a. Open PSADMIN.
   b. Enter *1* for Application Server and press Enter.
   c. Enter *1* for Administer a Domain and press Enter.
   d. Choose a domain from the list and press Enter.
   e. Choose *4* for Configure the Domain and press Enter.
   f. Enter *Y* to shut down the domain.
   g. Enter *Y* to change the configuration values.
   h. Press Enter to scroll through the file and accept the current settings until you reach the following section:

      ```
      Values for config section - PSAPPSRV
      ```

2. Enter *Y* and press Enter to change values in the section.
3. Navigate to the Tuxedeo Queue Size parameter. To do so, press Enter to scroll through the list and accept the current values. When you reach the Tuxedo Queue Size parameter enter a value.

   A value of *0* (zero) disables Tuxedo queue threshold determination and usage.

   Based on your environment, a value of *-1* sets the queue size to the following default values:

   - Windows: 65535.
   - AIX: 4000000.
   - Solaris: 65535.
   - HP: 65535.
4. Press Enter to scroll through the remaining sections and accept the current settings.

   PSADMIN will process the changes and then load the new configuration.

5. Boot the domain.

See Also

Enterprise PeopleTools 8.50 PeopleBook: System and Server Administration, "Using PSADMIN Menus"

Chapter 5

Managing Integration Gateways

This chapter provides an overview of integration gateway configuration and discusses how to:

- Administer integration gateways.
- Access gateway setup properties.
- Set Oracle Jolt connection properties.
- Use the integrationGateway.properties file.
- Encrypt passwords.
- Configure security and general properties.
- Mask gateway log elements.
- Configure integration gateways for load balancing.
- Refresh integration gateway properties.
- Bypass integration engines to send messages.

Understanding Integration Gateway Configuration

This section discusses:

- Integration gateway versions and application server versions.
- Local gateway compatibility.
- Types of integration gateway configuration.

Integration Gateway Versions and Application Server Versions

Local and remote integration gateways must be at the same or higher version as the application servers with which they communicate.

Out of the box, PeopleTools 8.50 is delivered with a PeopleTools 8.50 integration gateway and a PeopleTools 8.50 application server, thus meeting this requirement

However, situations may arise where your integration environment is comprised of PeopleSoft systems running different versions of PeopleTools, or your integration partners are running different versions of PeopleTools. When this is the case, you must ensure that the integration gateway is at the same or higher version as the application server with which it communicates or integrations will fail.

The following list describes several compatible integration gateway/application server version combinations:

- You are using a PeopleTools 8.50 integration gateway to communicate to a PeopleTools 8.50 application server.
- You are using a PeopleTools 8.50 integration gateway to communicate to a PeopleTools 8.49 or earlier application server.
- You are using a PeopleTools 8.50 remote integration gateway to communicate with a PeopleTools 8.50 or earlier application server.

The following list describes several incompatible integration gateway/application server version combinations. Integrations will fail with these combinations:

- You are using a PeopleTools 8.49 or earlier integration gateway to communicate with a PeopleTools 8.50 application server.
- You are using a PeopleTools 8.49 remote integration gateway to communicate with a PeopleTools 8.50 application server.

Local Gateway Compatibility

Because database administrator passwords and gateway keystore passwords are encrypted in the current PeopleTools release, the local gateway specified by a node in the current release of PeopleSoft Integration Broker must be from PeopleTools 8.41 or later to support encryption. If you upgrade PeopleTools and the integration engine from a release earlier than PeopleTools 8.41, you must also upgrade the local gateway.

Note. The current release of the integration gateway works with nodes that use PeopleTools 8.4x PeopleSoft Integration Broker.

Types of Integration Gateway Configuration

An integration gateway requires several types of configuration:

Security configuration You implement PeopleSoft Integration Broker security in several ways, including installing digital certificates on the gateway's web server and on the gateway itself to support Secure Sockets Layer (SSL) encryption. To implement encryption, you should complete the certificate installation before continuing with the gateway configuration in this chapter.

Once the gateway's digital certificates are installed, you must enter several configuration parameters in the Integration Gateway Certificates Section of the integrationGateway.properties file. The parameters you must set are the certificate alias name, the certificate alias password, the path to the keystore, and the keystore password.

See Chapter 11, "Setting Up Secure Integration Environments," Installing Digital Certificates for SSL Encryption on Oracle WebLogic, page 178.

General configuration This includes settings for the gateway version, class location, general communication parameters, node connection parameters, message and error logging, and gateway type and location. Most of these settings are entries in the integrationGateway.properties file, but you set a few of them in the Gateways component.

Connector-specific configuration The number of configuration settings and where they're applied depend on the connector. You configure most of the target connectors delivered with PeopleSoft Integration Broker by using the Gateways component, but some require settings in the integrationGateway.properties file. A few require settings in both environments.

Note. You can override some target connector properties for an individual node.

The Gateways Component

Once the gateway has been installed, you use the Gateways component (IB_GATEWAY) to make it accessible to any node that uses it for messaging. You can also use it to override the gateway's default connector properties for individual nodes without having to directly edit the integrationGateway.properties file on the gateway machine.

See Chapter 5, "Managing Integration Gateways," page 29.

Minimum Integration Gateway Setup Requirements

The minimum setup requirement to run an integration gateway are:

1. Specify the gateway URL.
2. Specify the Oracle Jolt connection string properties to enable communication with each PeopleSoft Integration Broker node that will be involved in an integration that uses a gateway.
3. Set and encrypt the keystore password.

See Also

Chapter 3, "Using the Integration Broker Quick Configuration Page," page 11

Chapter 5, "Managing Integration Gateways," Setting SSL Encryption Security Properties, page 46

Administering Integration Gateways

This section discusses how to:

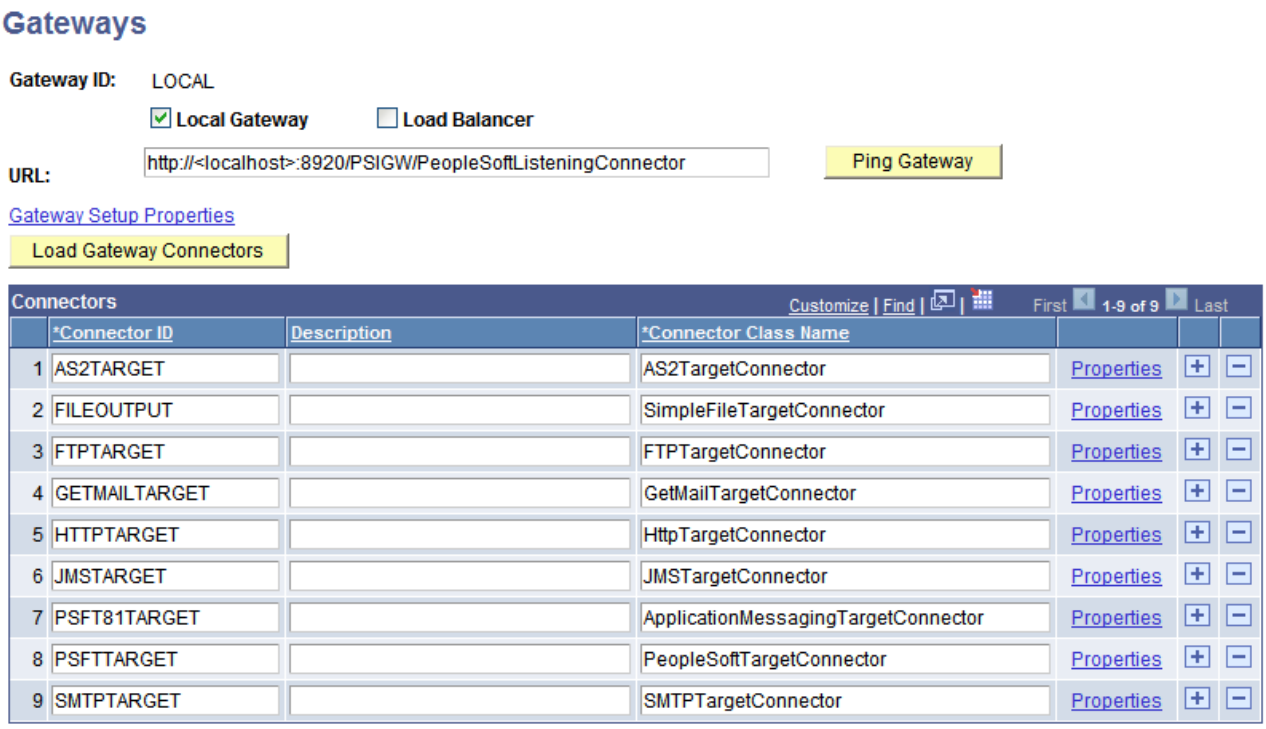
- Define integration gateways.
- Ping integration gateways.
- Register installed target connectors.
- Refresh the gateway properties.
- Edit available connector properties.

Defining Integration Gateways

Use the Gateways page (IB_GATEWAY) in the Gateways component (IB_GATEWAY) to specify the location of the gateway, update configuration settings, and register target connectors to be used with the gateway.

Note. A default local gateway definition is automatically created upon installation. If you plan to use only the local gateway, you do not need to create a new definition; however, you still must configure the gateway.

To access the Gateways page, select PeopleTools, Integration Broker, Configuration, Gateways. The following example shows the Gateways page:



Gateways page with the URL defined and target connectors loaded

To define and configure a gateway:

1. Access the Gateways page (select PeopleTools, Integration Broker, Configuration, Gateways).

   - Click Search, and select an existing gateway definition.

     The Gateways page appears, displaying the gateway definition.

     Note. The default ID for the delivered local gateway is *LOCAL*.

   - Add a new value, enter an integration gateway ID, and click Add.

     The Gateways page appears.

2. (Optional.) Select *Local Gateway* to designate the gateway as local.

   Each PeopleSoft Integration Broker node requires exactly one local gateway, which is the application's first point of contact with other PeopleSoft applications, third-party systems, Integration Broker hubs, and remote gateways.

   Note. You must open the definition of the designated local gateway and clear the Local Gateway check box before you can select that check box in another definition.

   ---

3. Enter the gateway URL for the selected gateway's PeopleSoft listening connector.

   Specify the URL with the format:

   http://*machinename:port*/PSIGW/PeopleSoftListeningConnector

   In this case, *machinename:port* is the machine name and port, host name, or IP address of the web server hosting the gateway.

   By default the port number is *80* for HTTP and *443* for HTTPS. If using the default port number, you do not need to specify it in the URL.

   For HTTPS, the URL should start with *https.*

   The integration gateway URL is case sensitive.

   The gateway uses the PeopleSoft listening connector to receive service operations from an integration engine node or a remote gateway.

4. (Optional.) To load the delivered target connectors, click the Load Gateway Connectors button.

   You can load the delivered target connectors at this point, or at a later time.

5. Save the gateway definition.

6. Click the Gateway Setup Properties link to configure additional gateway settings and connector properties.

See Also

Chapter 12, "Tuning Messaging System Performance," Configuring Integration Gateways for Load Balancing, page 261

Pinging Integration Gateways

Use the Gateways page to ping an integration gateway to verify that it is running. Before you ping an integration gateway, you must define the gateway URL.

To ping an integration gateway:

1. Access the Gateways page (select PeopleTools, Integration Broker, Configuration, Gateways).

2. Select the integration gateway to ping.

   The Gateways page appears.

3. Click the Ping Gateway button.

If the ping is successful a PeopleSoft Listening Gateway page appears that displays the PeopleTools release and a status of *Active.*

Loading Target Connectors

The Connectors grid on the Gateways page lists the target connectors registered with the current gateway. Initially, none of the delivered connectors are loaded and the grid is empty. You can load target connectors automatically by introspection or manually by entering information in the grid.

Note. You typically load and configure the gateway target connectors only when you configure a new gateway or install a new connector.

Loading Connectors by Introspection

If the connector was delivered with the PeopleSoft application or developed using the PeopleSoft Integration Broker Connector Software Development Kit (SDK), you can easily load it with the PeopleSoft Integration Broker connector introspection feature. Before you can register a new connector, you must install it.

See Appendix B, "Using the Integration Broker Connector SDK," page 301.

To load connectors by introspection:

1. Access the Gateways page (select PeopleTools, Integration Broker, Configuration, Gateways).
2. Click the Load Gateway Connectors button to trigger introspection for the current gateway.

PeopleSoft Integration Broker examines the properties of all installed target connectors and loads those properties into the gateway definition. All the connectors appear in the Connectors grid, and the properties of each connector are updated to reflect its current state.

Note. The introspection never overrides existing information. It adds only missing information, so manually edited values are not affected. If you modified a connector, new and modified properties are loaded and do not interfere with existing properties.

Loading Connectors Manually

To load and configure a connector manually, you enter the connector ID, connector class name, and property information that's hard-coded in the connector. This information is provided by PeopleSoft for all delivered connectors; information about connectors from any other source must be provided by that source.

To load a new connector manually:

1. Access the Gateways page (select PeopleTools, Integration Broker, Configuration, Gateways).
2. Add a new row in the Connectors grid.
3. Enter the ID for the new connector.
4. Enter the connector class name.
5. Click Properties to edit the connector's properties.

See Also

Appendix A, "Using the Delivered Listening Connectors and Target Connectors," page 267

Appendix B, "Using the Integration Broker Connector SDK," page 301

Editing Connector Properties

Node-level target connector properties represent parameters that can be used by the connector. These properties are hard-coded in the connector class. The Connector Properties page (IB_CONNPROP) lists all of a connector's available properties and their values. When you specify a connector in a node definition, only the properties that you are required to set and specify display.

Note. Available connector properties are automatically entered on the Connector Properties page when you register the connector.

Each property entry is defined by a combination of property ID and property name, both of which must already exist in the connector class. A single connector can handle service operations that adhere to different header formats, communication protocols, or other requirements. You can represent these variations on the Connector Properties page by entering multiple instances of the properties used, each with a different value.

Warning! Do not add new properties to any of the delivered connectors, as doing so requires changes to the delivered Java connector programs. Add connector properties only for custom connectors you have created.

The following example shows the Connector Properties page:

Gateways

Connector Properties

Gateway ID LOCAL

Connector SMTPTARGET

Properties Customize | Find | First 1-8 of 8 Last

Properties | Data Type / Description

	*Property ID	*Property Name	Required	Value	Default
1	HEADER	Content-Type	☐	text/plain	☐
2	HEADER	Content-Type	☐	text/html	☐
3	HEADER	sendUncompressed	☑	Y	☑
4	HEADER	sendUncompressed	☑	N	☐
5	SMTPTARGET	BCC	☐		☐
6	SMTPTARGET	CC	☐		☐
7	SMTPTARGET	DestEmailAddress	☑		☑
8	SMTPTARGET	SourceEmailAddress	☑		☑

Properties tab for the SMTP target connector

To add a new property instance:

1. Access the Connector Properties page (select PeopleTools, Integration Broker, Configuration, Gateways to display the Gateways page).
2. In the Connectors section locate the row that lists the target connector with which you want to work, and click the Properties link at the end of the row. The Connector Properties page displays.
3. Select a Property ID.

   Available property IDs are specific to the connector that you're configuring.
4. Select a Property Name.

   The available property names are specific to the property ID that you selected.
5. If the property is required for the connector to work properly, select the Required check box.

   All instances of a property (that is, all identical property ID and property name combinations) should have the same Required status.
6. Enter an appropriate value for the property instance.

   Appropriate values might come from PeopleSoft, from the connector's developer, or from your own experience and requirements.

7. (Optional.) Select the Default check box.

   When you specify the connector in a node definition, only properties marked as both required and default appear automatically on the Connectors page of the Node Definitions component.

   Note. In most cases, only one instance (value) of a required property should be used by a given node; however, you might designate multiple values as default so that they all appear. Keep in mind which properties can be used with multiple values and which ones require a single value.

8. Save the properties.

9. Click OK.

   The Gateways page appears.

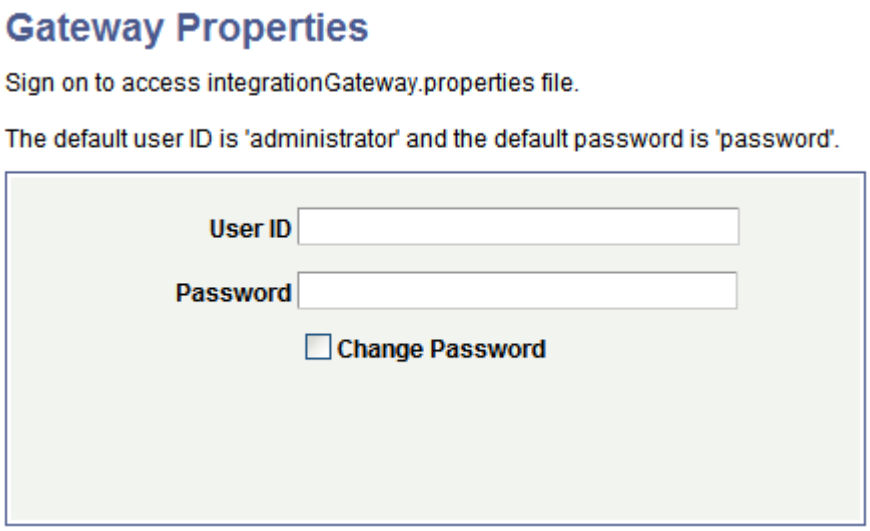
Accessing Gateway Setup Properties

This section discusses how to access the integration gateway set up properties.

You can access the gateway setup properties from the Quick Configuration page or from the Gateways page.

- To access gateway setup properties from the Quick Configuration page, select PeopleTools, Integration Broker, Configuration, Quick Configuration. The Quick Configuration page appears. Click the Advanced Gateway Setup link.
- To access gateway setup properties from the Gateways page, select PeopleTools, Integration Broker, Configuration, Gateways. The Gateways page appears. Click the Gateway Setup Properties link.

The Gateway Properties signon page (IBGWSIGNON) appears as shown in the following example:



Gateway Properties sign in page.

The default user ID is *administrator* and the default password is *password.* Check the Change Password box to change the default password.

Note. You should change the default password as soon as possible.

You can reset the password in the gatewayUserProfile.xml file located in <PIA_HOME>\webserv\<DOMAIN>\applications\peoplesoft\PSIGW.war\WEB-INF. The password you enter in the gatewayUserProfile.xml file must be encrypted. You can use the PSCipher utility to encrypt the password.

After you successfully enter the user ID and password, the PeopleSoft Node Configuration page displays where you specify information about how to connect to nodes and access the integrationGateway.properties file to establish additional gateway settings.

Setting Oracle Jolt Connection Properties

The integration gateway communicates with PeopleSoft application server nodes using Oracle Jolt connections.

Understanding Oracle Jolt Connection Properties

This section discusses setting Oracle Jolt connection string properties using the PeopleSoft Node Configuration page. Setting these properties in the integrationGateway.properties file is discussed later in this section.

The PeopleSoft Node Configuration page (PSGTWPROPS_SEC) provides grids for defining Oracle Jolt connection properties for unknown (default) and known nodes. When you save the properties you set on this page, they are written to the integrationGateway.properties file. To edit or define these properties in the future, you can use the PeopleSoft Node Configuration page or the integrationGateway.properties file.

Connection Settings When Target Nodes are not Known

Within any inbound message, the integration gateway requires only the names of the message and the requesting node. If the message is sent by a PeopleSoft Integration Broker system, it also includes the name of the target node. The gateway searches the integrationGateway.properties file for the Jolt connect string properties for the specified target node, so it can properly direct the message.

However, the integration gateway cannot determine the target node in the following cases:

- The Jolt connect string settings for the specified target node are missing from the integrationGateway.properties file.
- The message format does not include a To node specification.

To handle these cases, you can specify a default application server to handle the message if no valid target node can be determined.

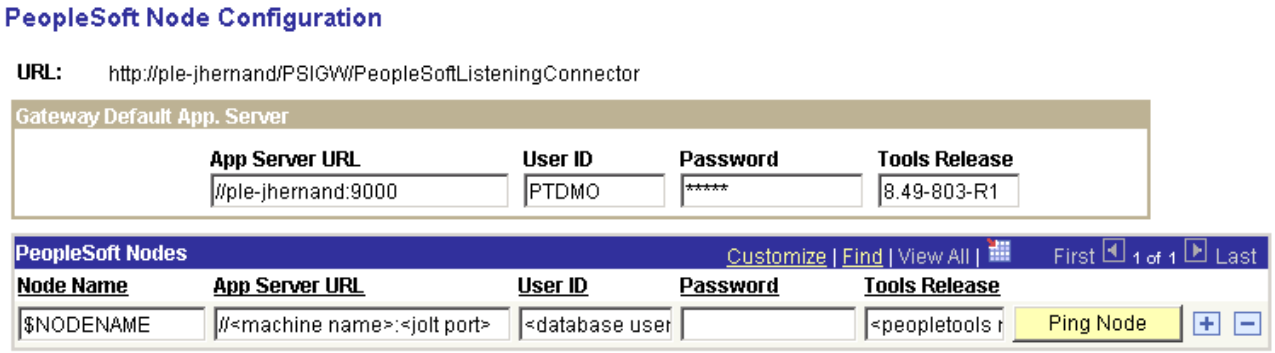
Connection Settings for Known Target Nodes

You must set four Oracle Jolt connect string properties for each PeopleSoft Integration Broker application server node with which the integration gateway communicates. The gateway uses this information to access each node's database through a Oracle Jolt connection with its PeopleSoft target connector.

Note. These properties apply only to communications that don't cross a firewall and for which the gateway uses the PeopleSoft target connector.

Setting Oracle Jolt Connection String Properties

The PeopleSoft Node Configuration page provides a grid for setting Oracle Jolt connection string properties for unknown (default) target nodes and known target nodes.



PeopleSoft Node Configuration page

To access the PeopleSoft Nodes Configuration page, select PeopleTools, Integration Broker, Configuration, Gateways. The Gateways page appears. Click the Gateway Setup Properties link. Enter the gateway user ID and password and click the OK button. The PeopleSoft Node Configuration page appears.

To define properties for unknown nodes use the Gateway Default Application Server grid on the PeopleSoft Node Configuration page. To define properties for known nodes use the PeopleSoft Node grid on the PeopleSoft Node Configuration page.

Note. Setting Oracle Jolt string connection properties for unknown nodes is optional.

App Server URL (Application Server URL in the Gateway Default App Server Section)

Enter the machine name and Oracle Jolt port number of the default application server to use if no valid target node can be determined.

To determine the Jolt port of the application server, check the JOLTListener section in the psappsrv.cfg file. The file is located in <PS_CFG_HOME>\appserv\<DOMAIN_NAME>.

App Server URL (Application Server URL in the PeopleSoft Nodes Section)

Enter the machine name and Oracle Jolt port number of the default application server to use if no valid target node can be determined.

Note. To determine the Jolt port of the application server, check the JOLTListener section in the psappsrv.cfg file. The file is located in <PS_CFG_HOME>\appserv\<DOMAIN_NAME>.

Node Name

Enter name of the PeopleSoft node with which the integration gateway is to communicate.

User ID	Enter the user ID that you defined when you created the application server domain.
Password	Enter the UserPswd that you defined when you created the application server domain. PeopleSoft Integration Broker will automatically encrypt this password entry.
Tools Release	Enter PeopleTools version number installed on the application server. Limit the number you enter to two decimal places. For example, *8.50.* If you are installing a patch build, include the patch number. For example, if you are installing PeopleTools 8.50 patch build 3, enter the following: *8.50.03*

The properties and values you set in the PeopleSoft Node Configuration page are located in the DELIVERED CONNECTOR CONFIGURATION Section of the integrationGateway.properties file.

The properties you set for unknown nodes are in the subsection ## JOLT connect string setting for optional Default Application Server. The properties you set for known nodes are in the subsection ## JOLT connect string settings for Application Server(s) with known NODENAMEs.

See Also

Chapter 5, "Managing Integration Gateways," Accessing Gateway Setup Properties, page 38

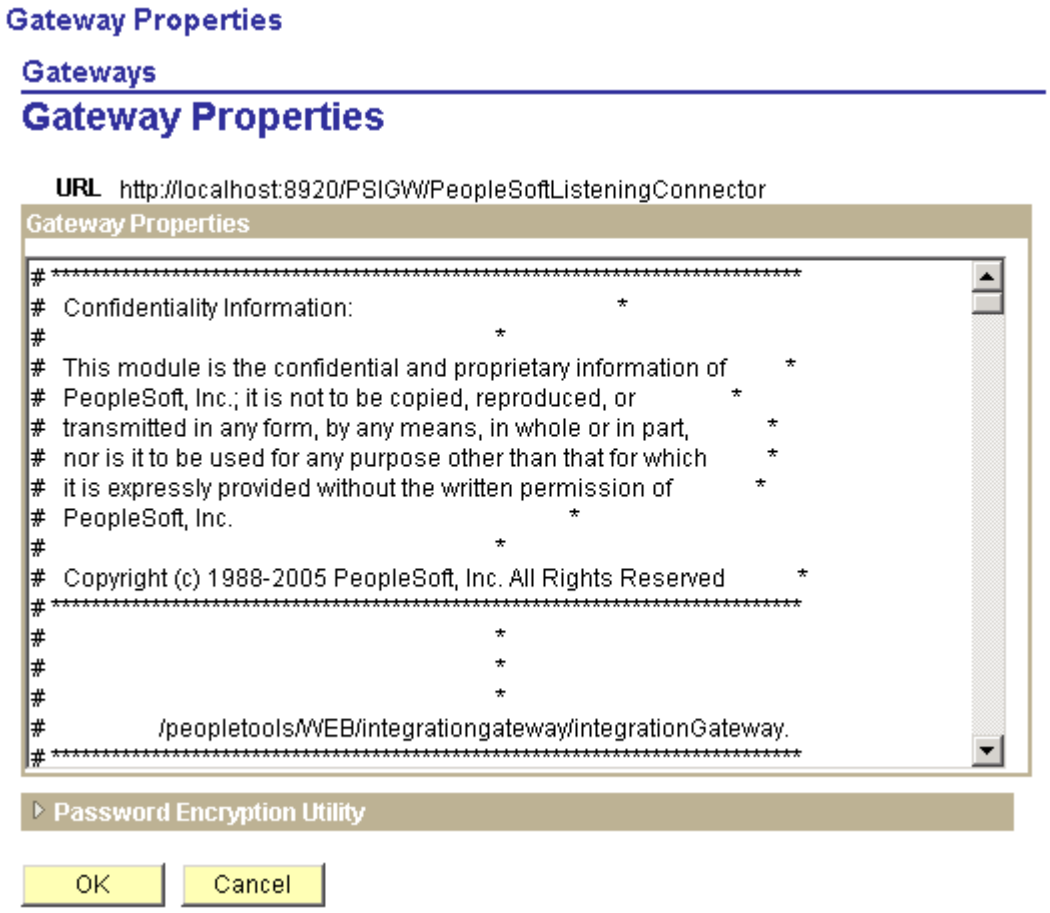
Chapter 5, "Managing Integration Gateways," Using the integrationGateway.properties File, page 41

Chapter 5, "Managing Integration Gateways," Configuring Security and General Properties, page 46

Using the integrationGateway.properties File

To establish settings for the integration gateway and its delivered connectors, you use the integrationGateway.properties file.

The integrationGateway.properties file is a text file.



Gateway Properties page

The property settings in the file are stored as name-value pairs in labeled sections, and the lines are commented out using the pound sign (#). Here's an example of a commented-out property setting:

```
#ig.isc.userid=MYUSERID
```

Accessing the integrationGateway.properties File

You can access and edit the integrationGateway.properties file using the Gateways component in the Pure Internet Architecture or using the text file located in the <PIA_HOME>\webserv directory.

Understanding Accessing the integrationGateway.properties File

Most integration systems are configured such that the application server, integration gateway, and PeopleSoft Pure Internet Architecture are running the same PeopleTools versions.

However, there may be some instances where the integration system is configured such that there is a shared integration gateway working with application servers on different versions of PeopleTools. If this is the case, you must access and edit the integrationGateway.properties in one of the following ways:

- Manually edit the text file located in the <PIA_HOME> directory. Use the PSCipher Java utility to generate encrypted passwords.

  Using the PSCipher Java utility is described elsewhere in this chapter.

  See Chapter 5, "Managing Integration Gateways," Encrypting Passwords Using the PSCipher Java Utility, page 45.

- Access and edit the properties file via the PeopleSoft Pure Internet Architecture in the Gateways component. You must log into the PeopleSoft Pure Internet Architecture that is installed on the application server that is running the same PeopleTools release as the integration gateway. If you use this method, you must ensure that copies of the same psvault key file are installed on all application servers and gateway/web server in the configuration.

  The psvault key file is discussed elsewhere in PeopleBooks.

  See *Enterprise PeopleTools 8.50 PeopleBook: Security Administration*, "Encrypting Text With PSCipher."

Accessing the integrationGateway.properties File in the Pure Internet Architecture

Access to the integrationGateway.properties file using the PeopleSoft Pure Internet Architecture is password-protected. You can access the file using the Integration Broker Quick Configuration page or the Gateways component.

To access the integrationGateway.properties file using the Integration Broker Quick Configuration page:

1. Select PeopleTools, Integration Broker, Configuration, Quick Configuration.

   The Integration Broker Quick Configuration page displays.

2. Click the Advanced Gateway Setup link.

   The Gateway page displays.

3. Click the Gateway Setup Properties link.

   The Sign on to access the integrationGateway.properties file box displays.

4. Enter the user ID and password and click the OK button.

5. Click the Advanced Properties Page link.

To access the integrationGateway.properties file using the Gateways component:

1. Select PeopleTools, Integration Broker, Configuration, Gateway.

2. Select a gateway with which to work.

3. Click the Gateway Setup Properties link.

   The Sign on to access the integrationGateway.properties file box displays.

4. Enter the user ID and password and click the OK button.

5. Click the Advanced Properties Page link.

The Gateway Properties page also provides access to the Password Encryption Utility and you can encrypt passwords required in the integrationGateway.properties file directly from that page.

Accessing the integrationGateway.properties File in the <PIA_HOME> Directory

The integrationGateway.properties file is located in the following path in the PeopleSoft home directory:

<PIA_HOME>\webserv\<DOMAIN>\applications\peoplesoft\PSIGW.war\WEB-INF\integrationGateway.properties.

When you access the integrationGateway.properties file in directly in the <PIA_HOME> directory, you must restart the PeopleSoft Pure Internet Architecture for the changes to take effect.

See Also

Chapter 5, "Managing Integration Gateways," Loading Target Connectors, page 35

Chapter 5, "Managing Integration Gateways," Encrypting Passwords, page 44

Entering Values in the integrationGateway.properties File

When entering values in the integrationGateway.properties file that contain paths, you must use either double-backslashes ("\\") or forward slashes ("/") as path separators.

Note. Do not use backslashes ("\") as path separators for directory names in the integrationGateway.properties file. Backslashes are misinterpreted as escape characters by the Java processes that access the file.

To correctly specify a path in the integrationGateway.properties file, you must use either double backslashes ("\\") or single forward slashes ("/") as separators; for example:

```
ig.transform1.XSL=C:\\XSLProgs\\MyTransform.xsl
ig.transform1.XSL=C:/XSLProgs/MyTransform.xsl
ig.transform1.XSL=/usr/xsls/MyTransform.xsl
```

Note. The one exception to this is when entering path separators for EIP test automation properties. When working with those properties you must enter path separators as backslashes.

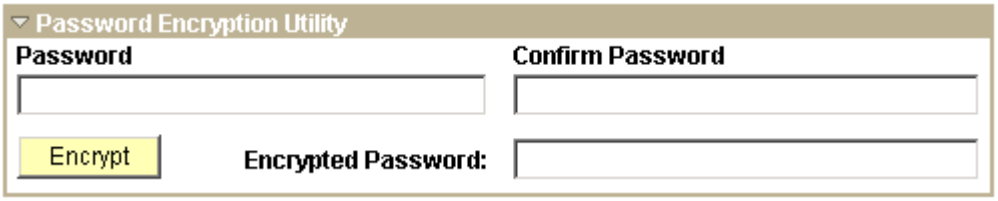
Encrypting Passwords

The integration gateway properties file and target connectors feature required and optional passwords. All passwords must be encrypted.

PeopleSoft provides an encryption utility, PSCipher, that you can use to encrypt passwords. You can access the utility from the PeopleSoft Pure Internet Architecture or from a Java utility.

Encrypting Passwords in the PeopleSoft Pure Internet Architecture

The Password Encryption Utility dialog box displays in areas where required or optional passwords are specified.



Password Encryption Utility dialog box

To encrypt a password using the Password Encryption Utility:

1. On the page where you are working, click the Password Encryption Utility arrow to display the dialog box.
2. In the Password field, enter a password.
3. In the Confirm Password field, enter the password again.
4. Click the Encrypt button. The encrypted password displays in the Encrypted Password field.
5. From the Encrypted Password field, cut the encrypted password and paste it into the appropriate location.

Encrypting Passwords Using the PSCipher Java Utility

You launch the PSCipher utility from the <PIA_HOME> directory.

To encrypt a password:

1. Launch the PSCipher.bat file in the <PIA_HOME>\webserv\<DOMAIN>\bin directory.
2. If using UNIX, change the script file's permissions so that you can execute it.
3. Execute the script file with your password as an argument.

   The utility returns the encrypted password as a string.

   - On a Windows machine, enter:

     ```
     pscipher MYPASSWORD
     ```

   - On a UNIX machine, enter:

     ```
     PSCipher.sh MYPASSWORD
     ```

4. Copy the encrypted string and paste it into the appropriate location.

Configuring Security and General Properties

This section discusses how to:

- Set SSL encryption security properties.
- Specify the gateway version.
- Specify the gateway class location.
- Set general connection properties.
- Set logging properties.
- Set DTD validation properties.
- Set Oracle Jolt session pooling parameters.
- Set the namespace for generic SOAP faults.

Setting SSL Encryption Security Properties

You can implement several types of messaging security with PeopleSoft Integration Broker. One type is SSL encryption, which applies digital certificates from two keystores to encrypt inbound and outbound messages, respectively. The integration gateway manages the certificates in the keystore that supports outbound messaging.

You must first install the certificates in the keystore. Then you set the security properties, which you can find in the integrationGateway.properties file section labeled *Integration Gateway CERTIFICATE Section.*

Warning! Integrations will fail if you do not set the path to the keystore using the secureFileKeystorePath property and enter an encrypted keystore password for the secureFileKeystorePasswd property.

You must set the following properties in integrationGateway.properties so that the gateway can access the SSL encryption certificates.

Property	*Description*
ig.certificateAlias	Enter the name that you provided to identify the encryption key pair that you generated for the keystore on which the gateway's public key certificate is based.
ig.certificatePasswd	Enter the password that you provided for the encryption key pair that you generated for the keystore. The certificate password must be encrypted. See Chapter 5, "Managing Integration Gateways," Encrypting Passwords, page 44.

Property	*Description*
secureFileKeystorePath	Enter the full path and file name of the gateway keystore file, which is located in the web server directory structure. The path is <PIA_HOME>\webserv\<DOMAIN>\keystore.
secureFileKeystorePasswd	Enter the keystore password. The default value is *password.* In a production system, the keystore password should be changed to a unique, non-trivial value. This password must be encrypted. See Chapter 5, "Managing Integration Gateways," Encrypting Passwords, page 44.

See Also

Chapter 11, "Setting Up Secure Integration Environments," page 155

Specifying the Gateway Version

The gateway version property, ig.version, indicates the version of PeopleTools from which the integration gateway is installed.

The integration gateway version must be the same or higher version than the version of the application server with which you want to communicate. For example, you can use a PeopleTools 8.50 integration gateway to communicate with a PeopleTools 8.50 application server or to communicate with a PeopleTools 8.47 application server.

Integration gateways cannot communicate with application servers on higher versions. For example a PeopleTools 8.47 integration gateway cannot directly communicate with a PeopleTools 8.50 application server. If this kind of communication is required, then the communication should be setup where the 8.47 gateway is set up as a remote gateway.

The version property is located in the integrationGateway.properties file in the section labeled Integration Gateway VERSION Section. Specify the version as follows:

```
ig.version=version_number
```

where *version_number* is the version of PeopleTools with two decimal places; for example, *8.50.*

Setting General Connection Properties

This section discusses:

- Default connector properties.
- Node-specific Oracle Jolt connect string properties.
- Default Oracle Jolt connect string properties.

The general connection properties include default connector properties and Oracle Jolt connect strings for nodes that designate this gateway as their local gateway. You can find these properties in the section of the integrationGateway.properties file labeled *DELIVERED CONNECTOR CONFIGURATION Section.*

Default Connector Properties

Property	Description
ig.connector.prefix	Identifies the universal resource indicator (URI) prefix added to any target connector name. This property instantiates the connector classes on the system. The default connector prefix is: `com.peoplesoft.pt.integrationgateway.targetconnector.` **Note.** Do not change this value.
ig.connector.defaultremoteconnector	Identifies the connector that the gateway uses to send messages to a remote gateway. The default value of this property is: `HttpTargetConnector` **Note.** Do not change this value.
ig.connector.ibtargetconnector	Identifies the connector that the gateway uses by default to send messages to a PeopleSoft Integration Broker application server node. The gateway uses this connector to link to the integration engine running on the node's application server. When the content of a message reaching the gateway doesn't specify a connector (this is often the case with third-party senders), the gateway automatically uses the connector specified by this property. The default value is: `PeopleSoftTargetConnector` **Note.** Do not change this value.

Default Oracle Jolt Connect String Properties

Within any inbound message, the integration gateway requires only the names of the message and the requesting node. If the message was sent by a PeopleSoft Integration Broker system, it also includes the name of the target node. The gateway searches the integrationGateway.properties file for the Jolt connect string properties for the specified target node, so it can properly direct the message.

However, the integration gateway cannot determine the target node in the following cases:

- The Jolt connect string settings for the specified target node are missing from the integrationGateway.properties file.
- The message format does not include a To node specification.

  This can include general HTTP calls to listening connectors other than the PeopleSoft listening connector.
- When using Send Master for testing purposes.

To handle these cases, you can specify a default target node for the gateway if no valid target node can be determined.

Use the default Jolt connect string properties:

```
#ig.isc.serverURL=//<machine name>:<jolt port>
#ig.isc.userid=<database user id>
#ig.isc.password=<database password>
#ig.isc.toolsRel=<peopletools release version>
```

Uncomment these four lines and enter values to designate a PeopleSoft Integration Broker node as the gateway's default (backup) target node. It typically is one of the nodes for which you already created node-specific Jolt connect string properties.

There's only one set of these default properties. They specify the same parameters as the node-specific properties, except that you don't include a node name; for example:

```
ig.isc.serverURL=//MYMACHINE:9000
ig.isc.userid=TOPDOG
ig.isc.password=VOBN5KcQZMg
ig.isc.toolsRel=8.50
```

See Chapter 5, "Managing Integration Gateways," Oracle Jolt Connect String Properties for Known Nodes, page 49.

Oracle Jolt Connect String Properties for Known Nodes

You must set four Oracle Jolt connect string properties for each PeopleSoft Integration Broker application server node with which the integration gateway communicates. The gateway uses this information to access each node's database through a Oracle Jolt connection with its PeopleSoft target connector.

Note. These properties apply only to communications that don't cross a firewall and for which the gateway uses the PeopleSoft target connector.

The integrationGateway.properties file contains a template for these properties:

```
ig.isc.$NODENAME.serverURL=//<machine name>:<jolt port>
ig.isc.$NODENAME.userid=<database user id>
ig.isc.$NODENAME.password=<database password>
ig.isc.$NODENAME.toolsRel=<peopletools release version>
```

For each node, make a copy of this template and replace *$NODENAME* with the name of the node definition. Enter appropriate values for each property as described in the following table:

Property	*Description*
ig.isc.$NODENAME.serverURL	Enter the URL of the application server node, consisting of the machine name and Oracle Jolt port; for example: `ig.isc.MYNODE.serverURL=//MYMACHINE:9000` **Note.** You can determine the Jolt port of the application server by examining the *JOLT Listener* section in the psappsrv.cfg file located in <PS_CFG_HOME>\appserv\<DOMAIN_NAME>.

Property	*Description*
ig.isc.$NODENAME.userid	Enter the UserID that you defined when you created the application server domain; for example: `ig.isc.MYNODE.userid=TOPDOG`
ig.isc.$NODENAME.password	Enter UserPswd that you defined when you created the application server domain. This password must be encrypted; for example: `ig.isc.MYNODE.password=VOBN5KcQZMg` See Chapter 5, "Managing Integration Gateways," Encrypting Passwords, page 44.
ig.isc.$NODENAME.toolsRel	Enter the version number of PeopleTools installed on the application server node to two decimal places; for example: `ig.isc.MYNODE.toolsrel=8.50`

Setting Logging Properties

This section discusses:

- General logging properties.
- Message logging properties.
- Error logging properties.

The logging properties specify parameters for logging messaging activity and errors. You can find these properties in the section of the integrationGateway.properties file labeled *LOGGING Section.*

General Logging Properties

Property	Description
ig.log.level	Enter a numeric value to specify the desired level of gateway logging and exception handling. Values are: • *−100:* Suppresses message logging. The property is preset to this value. • *−1:* Logs language exceptions only. • *1:* Logs language and standard exceptions. • *2:* Logs all errors and warnings. • *3:* Logs errors, warnings, and important information. This is the default if you don't specify a value for this property. • *4:* Log errors, warnings, and important and standard information. • *5:* Logs errors, warnings, and important, standard, and low-importance information. **Note.** Set the log level to *5* to capture the entire contents of incoming HTTP requests, including HTTP headers, in the integration gateway message log file.
ig.log.backgroundImage	Specify the background image to use when displaying error and message log documents. The image must be in jpg format. The default location and image name PSbackground.jpg. By default it is located in <PIA_HOME>\webserv\<DOMAIN>\applications\peoplesoft\PSIGW.war. Images in the default location don't require a path, but you can specify a full path to an image file in any other location.

Message Logging Properties

Property	Description
ig.messageLog.filename	Enter the full path and file name of an HTML file to use as a message log. This property is preset to <PIA_HOME>\webserv\<DOMAIN>\applications\peoplesoft\PSIGW.war\msgLog.html.
ig.messageLog.maxSize	Specify the maximum size of the message log, in kilobytes (KB). This property is preset to *10000*, or 10 megabytes (MB). When this limit is reached, the log is archived, and a timestamp is appended to the file name.

Property	Description
ig.messageLog.maxNbBackupFiles	Specify the number of archived files to keep on disk. Use the value *0* to retain all backed up files. This property is preset to *5*.

Error Logging Properties

Property	Description
ig.errorLog.filename	Enter the full path and file name of an HTML file to use as an error log. This property is preset to <PIA_HOME>\webserv\<DOMAIN>\applications\peoplesoft\PSIGW.war\errorLog.html.
ig.errorLog.maxSize	Specify the maximum size of the error log in kilobytes (KB). This property is preset to *1000*, or 1 MB. When this limit is reached, the log is archived, and a timestamp is appended to the file name.
ig.errorLog.maxNbBackupFiles	Specify the number of archived error files to keep on disk. Use the value *0* to retain all backed up files. This property is preset to *5*.

See Also

Enterprise PeopleTools 8.50 PeopleBook: PeopleSoft Integration Broker, "Managing Error Handling, Logging, Tracing, and Debugging," Understanding Error Handling, Logging, Tracing and Debugging

Setting DTD Validation Properties

You can validate XML request messages and response messages against associated document type definitions (DTD) by enabling DTD validation on the integration gateway.

When you set the ig.dtdLookup property equal to *True* (default), request and response messages are validated against any associated DTD.

References to DTDs may be inline pointers to files or references to URLs.

When you set the ig.dtdLookup property equal to *False*, no validation takes place—even if a DTD reference is supplied.

If the ig.dtdLookup property is removed or otherwise missing from the integrationGateway.properties file, the system responds as if the property is set to *True*, and request and response messages are validated against any associated DTD.

Setting Oracle Jolt Session Pooling Parameters

The integration gateway maintains a pool of jolt sessions to handle requests between itself and the integration engine. The integration gateway issues a jolt session from the pool, uses it for the connection, and then returns the session to the pool once it receives the response from the integration engine.

The number of sessions to maintain in the session pool is defined in the integrationGateway.properties file using the following property:

`ig.connection`

Set this property equal to the maximum number of sessions to maintain in the pool. The default value is *10*.

Setting the Namespace for Generic SOAP Faults

The system generates generic SOAP faults for framework-level errors, such as when it cannot find a routing for an integration.

To specify the namespace to use for generic SOAP faults, set the following property in the integrationGateway.properties file equal to the namespace to use:

`ig.GenericFaultNamespace`

Masking Gateway Log File Elements

This section provides and overview of masking gateway log file elements and discusses how to:

- Access the logfilter.properties file.
- Mask elements not contained in namespaces.
- Mask elements contained in namespaces.
- Mask attributes of elements.
- Mask child element names.
- Change the global mask message.
- Create custom mask messages.
- Disable gateway log masks.

Understanding Masking Gateway Log File Elements

You can mask, or hide, elements that appear in the integration gateway log files, thereby prohibiting sensitive information from displaying in the generated logs.

Note. The system applies gateway log masks and messages to both the integration gateway message log file (MsgLog.html) and the integration gateway error log file (ErrorLog.html).

Global and Custom Mask Messages

By default, all masked elements have a global mask message applied to them, whereby every element you mask is replaced with a standardized message. You can also create custom mask messages for specific elements. You can use a combination of global and custom mask messages.

The default global mask message is **** deleted for security purposes ****. You can change the global mask as you wish to a message that best suits your business needs.

Default Masks

Several gateway log masks are implemented by default. They include, but are not limited to, the following elements:

- WSSE password.
- NodePassword.
- ExternalUserPassword.
- XML format request with password.
- PSFT AuthToken.
- SAML-TokenData.

You can disable any of these masks in the logfilter.properties file.

logfilter.properties File

To mask and unmask gateway log file elements use the logfilter.properties file.

To use the file to specify the element names, attribute names, and element namespaces to mask. You can also use the file to change the global mask message and set up custom mask messages.

Note. After you make any changes to the logfilter.properties file, you must reboot the webserver for the changes to take effect.

Property Types

The following table lists the property types with which you can work in the logfilter.properties file:

Note. The examples provided in this section show property names appended with a number. These numbers are property indexes and are discussed elsewhere in this section.

Property	*Description*
AttributeName	Set this property equal to an attribute of an element to mask.

Property	*Description*
ElementName	Set this property equal to an element name to mask.
IsLeaf	Use this property to mask an element and all child tags of the element.
Namesapace	Use the Namespace property in conjunction with the ElementName property to specify the namespace of the element to mask.

Property Indexes

All properties in the logfilter.properties file are appended with an index number. Indexes group related properties and their values. The following example shows an excerpt from the logfilter.properties file and the ElementName.<index_number> naming scheme.

```
#IBInfo NodePassword
ElementName.2=NodePassword

#IBInfo ExternalUserPassword
ElementName.3=ExternalUserPassword

#XML format request with Password
ElementName.4=Password
```

You can use any number as an index number. Index numbers do not have to be used in sequence. Using the previous example, if you were to add a new element name to the file, you would not have to name it *ElementName.5.* You could use any number not already in use, such as *ElementName.72.*

Properties can appear in any order in the logfilter.properties file and do not have to appear in sequential index order. As an example, *ElementName.72* could appear first in the file, followed by *ElementName.3,* followed by *ElementName.1,* followed by *ElementName.12,* and so on.

Mask Variables

PeopleSoft Integration Broker provides the following mask variables:

Mask Variable	*Description*
GlobalReplaceWith	By default the system assigns the value of this variable to all asked elements. The default global mask message is: `***deleted for security purposes***`
ReplaceWith	Use this variable to override the global mask value for a specific element and set a custom mask message.

Accessing the logfilter.properties File

The logfilter.properties file is located in the following path in the PeopleSoft home directory:

```
<PIA_HOME>\webserv\<DOMAIN>\applications\peoplesoft\PSIGW.war\WEB-INF⇒
\logfilter.properties
```

Masking Element Names Not Contained in Namespaces

Use the ElementName property to mask an element name that is not contained in a namespace.

To mask an element name that is not contained in a namespace, enter the element name to mask in logfilter.properties file in the following format:

```
ElementName.<index_number>=<Element_to_mask>
```

Be sure to specify a unique index number.

An example of a mask for an element name is shown in the following example.

```
ElementName.1=NodePassword
```

If you are using the default global mask, the element appears as follows in the gateway log files:

```
<NodePassword>*** deleted for security purposes ***</NodePassword>
```

Masking Element Names Contained Within Namespaces

Use the ElementName property and the Namespace property to mask elements contained within namespaces.

To mask an element name contain within a namespace, enter the element name to mask and namespace in which it is contained in the logfilter.properties file in the following format:

```
ElementName.<index_number>=<Element_to_mask>
Namespace.<index_number>=<Namespace_that_contains_element>
```

The ElementName and Namespace properties must use the same unique index number.

The following example shows how to enter a mask for the Username element contained in a namespace:

```
ElementName.9=Username
Namespace.9=http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-⇒
secext-1.0.xsd
```

If you are using the default global mask, the element appears as follows in the gateway log files:

```
<Username>*** deleted for security purposes ***</Username>
```

Masking Attributes of Element Names

Use the ElementName property and the AttributeName property to mask an attribute of an element.

To mask an attribute of an element, enter the element name and attribute name in the logfilter.properties file in the following format:

```
ElementName.<index_number>=<Element_name>
AttributeName.<index_number>=<Attribute_of_element_to_mask>
```

The ElementName and AttributeName properties must share the same unique index number.

The following example show the default mask for the password from the requesting node of a PeopleSoft 8.1x system:

```
#8.1x from node password
ElementName.6=from
AttributeName.6=password
```

An example request before masking is:

```
<?xml version="1.0"?>
<request version="1.0">
    <from node="PSFT_HR" password="my_password"/>
    <to node="QE_LOCAL"/>
```

When the mask is applied, the request looks as follows:

```
<?xml version="1.0"?>
<request version="1.0">
    <from node="PSFT_HR" password="*** Deleted for security purposes ***"/>
    <to node="QE_LOCAL"/>
```

Masking Child Element Names

Use and set the IsLeaf property equal to *false* to mask an element and all child elements. By default, child tags are not masked.

To mask an element and all child elements, enter the element name and set the IsLeaf property in the logfilter.properties file in the following format:

```
ElementName.<index_number>=<Element_name_(and_child_elements)_to_mask>
IsLeaf.<index_number>=false
```

The ElementName and IsLeaf properties must use the same unique index number.

As an example an address element could contain street number, street name, city, state, and zip code tags, as shown in the following example:

```
<address>
  <streetnumber>4433</streetnumber>
  <street>Oracle Lane</street>
  <city>Pleasanton</city>
  <state>California></state>
  <zipcode>94588</zipcode>
</address>
```

The following example shows how to mask the address element and all children of the element:

```
ElementName.11=address
IsLeaf.11=false
```

If you are using the default global mask, the element appears as follows in the gateway log files:

```
<address>***deleted for security purposes***</address>
```

However, if you wanted to mask just one of the child elements such as zip code, you would do so as shown in the following example:

```
ElementName.11=zipcode
```

The following example shows how the zip code tag would appear in the gateway logs if using the default global mask:

```
<address>
  <streetnumber>4433</streetnumber>
  <street>Oracle Lane</street>
  <city>Pleasanton</city>
  <state>California></state>
  <zipcode>***deleted for security purposes***</zipcode>
</address>
```

Changing the Global Mask Message

The value of the GlobalReplaceWith variable located in the logfilter.properties file determines the default global mask message. The default value is:

```
GlobalReplaceWith=***deleted for security purposes***
```

You can change this value as necessary to suit your business needs by setting the GlobalReplaceWith variable equal to another value. For example:

```
GlobalReplaceWith=####  PeopleSoft Confidential Information  ####
```

Creating Custom Mask Messages

You can override the global mask message on an element-by-element basis by setting the RepalceWith variable equal to a custom mask message.

The format is:

```
ElementName.<index.number>=<Element_to_mask>
ReplaceWith.<index_number>=<Custom_mask_message>
```

The index number you set must be the same unique index number used for the element, namespace, and/or attribute entry.

The following code snippet shows an example of overriding the default global mask message with a custom message:

```
#PSFT AuthToken
ElementName.7=AuthToken
ReplaceWith.7=-->Proprietary Information<--
```

When the gateway logs are generated the mask for this element will look as follows:

```
<AuthToken>-->Proprietary Information<--</AuthToken>
```

The following code example was shown earlier in this section. It has been modified to show how to override the default global mask message with a custom message:

```
ElementName.9=Username
Namespace.9=http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-⇒
secext-1.0.xsd
ReplaceWith.9=**  Data removed per company security policy **
```

When the gateway logs are generated the mask for this element will appear as follows:

```
<Username>**  Data removed per company security policy **</Username>
```

Disabling Gateway Log Masks

You can disable a mask for any element by commenting out the mask data in the logfilter.properties file.

For example, the following sample mask entry could appear in the logfilter.properties file:

```
#Sample Mask Entry
ElementName.44=NodeName
Namespace.44=http://my_namespace.xsd
ReplaceWith.44=--->Confidential/Proprietary Information<---
```

To disable the entry comment out all lines as shown in the following example:

```
#Sample Mask Entry
#ElementName.44=NodeName
#Namespace.44=http://my_namespace.xsd
#ReplaceWith.44=--->Confidential/Proprietary Information<---
```

Refreshing Integration Gateway Properties

If you modify integration gateway properties by accessing and directly modifying the integrationGateway.properties file located in the <PIA_HOME> directory, you must restart the web server for the changes to take effect.

If you modify integration gateway properties in the PeopleSoft Pure Internet Architecture, any changes you make take effect when you save the changes or click the OK button. This includes changes you make to the integrationGateway.properties file, but only if you access the file through the PeopleSoft Pure Internet Architecture using the Gateways Properties page.

Bypassing Integration Engines to Send Messages

You can use the PeopleCode built-in functions ConnectorRequest and ConnectorRequestURL to send synchronous requests directly to the integration gateway, without any message processing taking place on the integration broker engine, thereby eliminating the need for transactions.

Note. ConnectorRequest and ConnectorRequestURL are for use with synchronous requests only.

To use any of these methods, the integration gateway must be configured and running.

When you use either of these functions, errors and messages are written to the integration gateway logs.

See Also

Enterprise PeopleTools 8.50 PeopleBook: PeopleSoft Integration Broker, "Sending and Receiving Messages," Generating and Sending Messages

Using the ConnectorRequest Built-In Function

The ConnectorRequest function enables you to build a message object and perform a POST or GET using any target connector. With this function, you use the Message object to populate connector values.

Response messages are returned unstructured in the *IB_GENERIC* message. The *IB_GENERIC* message is delivered out-of-the-box.

The following example shows using the ConnectorRequest function to perform a GET to obtain a stock quote.

```
Local XmlDoc &Output;

Local Message &MSG1, &MSG2;

&MSG = CreateMessage(OPERATION.QE_FLIGHTPLAN);

&MSG.IBInfo.IBConnectorInfo.ConnectorName = "HTTPTARGET";
&MSG.IBInfo.IBConnectorInfo.ConnectorClassName = "HttpTargetConnector";

&nReturn = &MSG.IBinfo.IBConnectorInfo.AddConnectorProperties
    ("Method", "GET", %HttpProperty);
&nReturn = &MSG.IBinfo.IBConnectorInfo.AddConnectorProperties
    ("URL", "http://finance.yahoo.com/d/quotes.txt/?symbols
   =PSFT&format=l1c1d1t1", %HttpProperty);

&MSG2 = %IntBroker.ConnectorRequest(&MSG);

&Output = &MSG2.GetXmlDoc();  // Get the data out of the message
```

Using the ConnectorRequestURL Built-In Function

The ConnectorRequestURL function enables you to use HTTP or FTP to perform a GET using a query string.

Based on the format of the string you provide, the integration gateway uses the HTTP target connector or FTP target connector to perform the GET.

Response messages are returned in a string.

Using ConnectorRequestURL with HTTP

The following example shows using the ConnectorRequestURL function to perform a GET to obtain a stock quote using HTTP.

```
&Output = %IntBroker.ConnectorRequestURL("http://finance.yahoo.com/d/quotes.txt/
?symbols=PSFT&format=l1c1d1t1");
```

Using ConnectorRequestURL with FTP

The syntax of the FTP URL is:

```
ftp://<user>:<password>@<host>:<port>/<url-path>;type=<typecode>
```

The following example shows using the ConnectorRequestURL function to perform a GET to obtain a stock quote using FTP.


```
&Output = %IntBroker.ConnectorRequestURL("ftp://qedmo:qedmo@ftp.globalsoft.com:
200/tmp/hello.xml;type=a");
```

Chapter 6

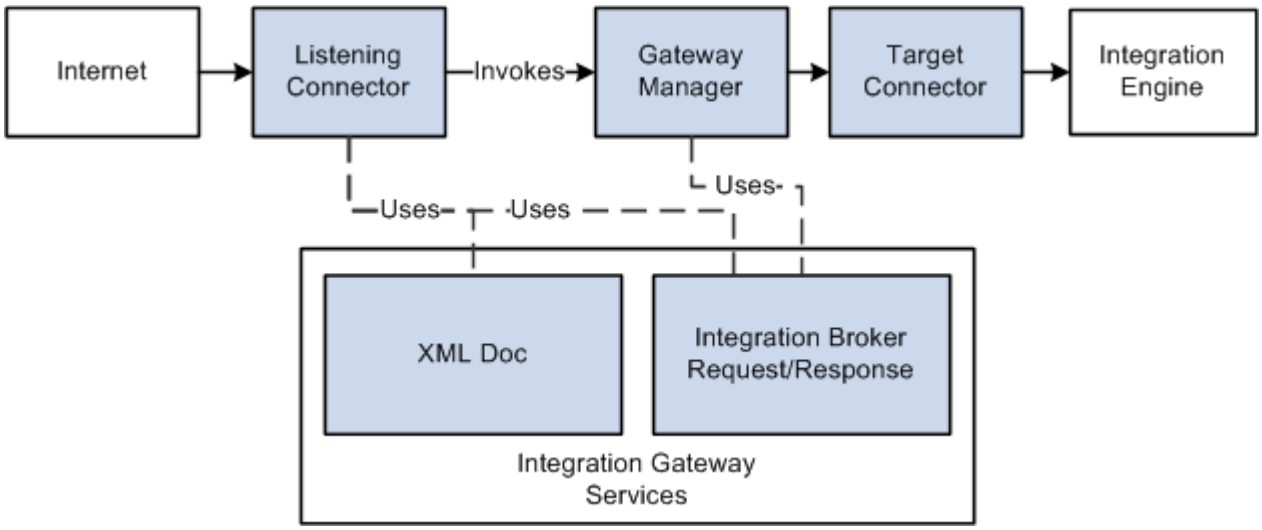
Using Listening Connectors and Target Connectors

This chapter discusses how to:

- Work with the PeopleSoft connectors.
- Work with the HTTP connectors.
- Work with the PeopleSoft services listening connector.
- Work with the PeopleSoft 8.1 connectors.
- Work with the Java Messaging Service (JMS) connectors.
- Work with the simple file target connector.
- Work with the File Transfer Protocol (FTP) target connector.
- Work with the AS2 connectors.
- Work with the Simple Mail Transfer Protocol (SMTP) target connector.

Understanding Listening Connectors

Listening connectors receive requests from integration participants, send them to the gateway manager, and deliver responses back to the integration participants. The following diagram shows the flow of an inbound request from an external system into the integration engine through a listening connector:



Request flow through a listening connector

PeopleSoft-Delivered Listening Connectors

PeopleSoft delivers several listening connectors with PeopleSoft Integration Broker that enable integration participants to communicate with the PeopleSoft system using a number of communication formats.

You send messages to a listening connector at a URL address derived from its class location on the gateway web server.

Note. The integration gateway provides a matching target connector for all connectors in the following table, except for the services listening connector. Although this chapter discusses each pair of listening and target connectors in a separate section, the use of a particular listening connector does not obligate you to use the corresponding target connector.

Connector	***Description***
PeopleSoft listening connector	In combination with the PeopleSoft target connector, this connector establishes the primary connection between a PeopleSoft application's integration engine and its local gateway. It receives requests from integration participants in the PeopleSoft internal messaging format. Third-party applications and PeopleSoft releases that don't include PeopleSoft Integration Broker should not send messages to this connector. See Chapter 6, "Using Listening Connectors and Target Connectors," Working With the PeopleSoft Connectors, page 70.
HTTP listening connector	This connector provides a web-standard method of communicating with the gateway. It accepts HTTP requests using the GET and POST methods. It also accepts secure HTTPS requests if SSL encryption is configured on the gateway's web server. See Chapter 6, "Using Listening Connectors and Target Connectors," Working With the HTTP Connectors, page 71.

Connector	*Description*
PeopleSoft 8.1 listening connector	This connector enables PeopleSoft 8.1x applications to communicate with the gateway using native Application Messaging technology. Third-party applications can send properly formatted 8.1x application messages to this connector. It also accepts secure HTTPS requests if SSL encryption is configured on the gateway's web server. See Chapter 6, "Using Listening Connectors and Target Connectors," Working With the PeopleSoft 8.1 Connectors, page 88.
JMS listening connector	This connector enables JMS provider systems to communicate with the gateway using standard JMS protocols. See Chapter 6, "Using Listening Connectors and Target Connectors," Working With the JMS Connectors, page 90.
AS2 listening connector	The AS2 listening connector enables you to receive request messages in AS2 format. See Chapter 6, "Using Listening Connectors and Target Connectors," Working With the AS2 Connectors, page 113.
PeopleSoft services listening connector	PeopleSoft Integration Broker uses the PeopleSoftServiceListeningConnector as an endpoint for all node transactions that you expose as WSDL. All PeopleSoft node transactions that you publish as WSDL have the following endpoint: http://<machine>/PSIGW/PeopleSoftServiceListeningConnector. See Chapter 6, "Using Listening Connectors and Target Connectors," Working With the PeopleSoft Services Listening Connector, page 86.

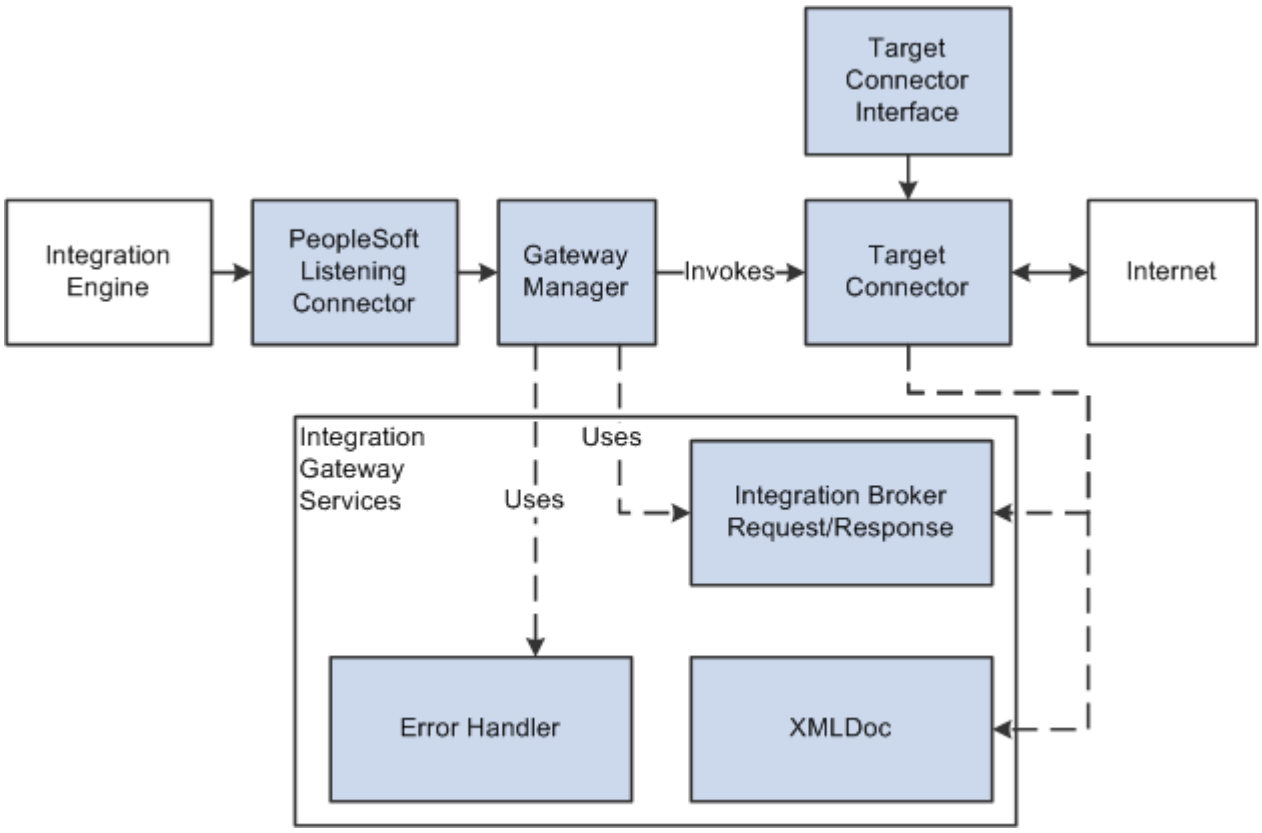
All of the delivered listening connectors that service HTTP requests run as servlets and are configured to run in Oracle WebLogic and IBM WebSphere web server environments. The delivered listening connectors that service HTTP requests are the PeopleSoft listening connector, the HTTP listening connector, and the PeopleSoft 8.1 listening connector.

Null Characters in Messages

The listening connectors delivered with PeopleSoft Integration Broker do not support null characters (ASCII value 00) as part of message field data. If a third-party application sends a message containing null characters, you must replace each instance of the null character with an acceptable substitute character, such as a space, before sending the message to the PeopleSoft system. Alternatively, you can modify the delivered listening connector to replace the null characters when it receives the message.

Understanding Target Connectors

Target connectors generate requests, send them to integration participants, wait for responses from participants, and deliver the responses back to the gateway manager, as shown in the following diagram:



Request and response flow through a target connector

The integration gateway invokes target connectors dynamically through the gateway manager. Target connectors adhere to a standard structure by implementing the target connector interface provided by the integration gateway. By implementing this interface, target connectors can take advantage of all gateway manager services.

Each target connector has an internal connector ID that you use when selecting the connector; for example, the connector ID for the simple file target connector is *FILEOUTPUT*.

PeopleSoft-Delivered Target Connectors

PeopleSoft delivers several target connectors with PeopleSoft Integration Broker that enable you to communicate with integration participants using a wide range of communication formats. The following table describes the delivered target connectors:

Connector	*Description*
PeopleSoft target connector	In combination with the PeopleSoft listening connector, this connector establishes the primary connection between a PeopleSoft application's integration engine and its local gateway. It sends requests to integration participants over a Oracle Jolt connection in the PeopleSoft internal messaging format. Use this connector to send messages only to PeopleSoft applications that use PeopleSoft Integration Broker. **Note.** Oracle Jolt is a Java-based interface that extends Oracle Tuxedo capabilities to the internet. The integration gateway uses it as the standard interface for communicating with integration engines through the PeopleSoft target connector. See Chapter 6, "Using Listening Connectors and Target Connectors," Working With the PeopleSoft Connectors, page 70.
HTTP target connector	This connector provides a web-standard method for the gateway to communicate with PeopleSoft and third-party applications. It sends HTTP requests using the GET and POST methods. It also sends secure HTTPS requests if SSL encryption is configured on the gateway. See Chapter 6, "Using Listening Connectors and Target Connectors," Working With the HTTP Connectors, page 71.
PeopleSoft 8.1 target connector	This connector enables the gateway to communicate with PeopleSoft 8.1x applications that use Application Messaging technology. It converts outbound messages to the Application Messaging native format. It also sends secure HTTPS requests if SSL encryption is configured on the gateway. See Chapter 6, "Using Listening Connectors and Target Connectors," Working With the PeopleSoft 8.1 Connectors, page 88.
JMS target connector	This connector enables the gateway to communicate with JMS provider systems using standard JMS protocols. See Chapter 6, "Using Listening Connectors and Target Connectors," Working With the JMS Connectors, page 90.
FTP target connector	This connector enables the gateway to transfer messages to an FTP server. It converts outbound messages to file data it can send using the FTP PUT command. You can also send messages over a secure FTP(S) protocol. In addition you can receive messages from FTP servers using the GET command. See Chapter 6, "Using Listening Connectors and Target Connectors," Working With the FTP Target Connector, page 107.
AS2 target connector	The AS2 target connector enables you to send messages in AS2 format. See Chapter 6, "Using Listening Connectors and Target Connectors," Working With the AS2 Connectors, page 113.

Connector	*Description*
SMTP target connector	With this connector, the gateway can send messages to an SMTP server using the PUT command. See Chapter 6, "Using Listening Connectors and Target Connectors," Working With the SMTP Target Connector, page 124.
Simple file target connector	With this connector, the gateway saves outbound messages as XML files. See Chapter 6, "Using Listening Connectors and Target Connectors," Working With the Simple File Target Connector, page 105.
GetMail target connector	This connector provides functionality specific to the PeopleSoft Multichannel Framework. See *Enterprise PeopleTools 8.50 PeopleBook: PeopleSoft MultiChannel Framework*, "Configuring the Email Channel."

Target Connector Properties

Most of the delivered target connectors have required and optional configuration properties that you set to control the connectors' behavior. Depending on the connector, you configure some of these properties in the integrationGateway.properties file or by using the Gateways component. You can specify values for connector properties in the following ways:

- *Gateway-level* target connector properties always have the same value for a given connector, regardless of which nodes or transactions use the connector.

  You specify the values of these properties in the integrationGateway.properties file.

- *Node-level* target connector properties can have different values for each default local node that uses a given gateway.

  Each node-level connector property is identified by a property ID and a property name. You specify default values for these properties in the Gateways component of each participating node.

  See Chapter 5, "Managing Integration Gateways," page 29.

When you create a node definition in the local database, you specify which gateway and target connector should be used to send messages to that node. In the node definition, you can supply values for the connector's node-level properties that override the defaults and apply only when sending messages to that node.

See Chapter 7, "Adding and Configuring Nodes," page 127.

When you define a routing definition, you can supply values for the connector's node-level properties to override the node definition's values and apply only when sending messages with that transaction.

See *Enterprise PeopleTools 8.50 PeopleBook: PeopleSoft Integration Broker*, "Managing Service Operation Routing Definitions," Overriding Gateway and Connector Properties.

You can set and override target connector properties at runtime using PeopleCode.

See *Enterprise PeopleTools 8.50 PeopleBook: PeopleSoft Integration Broker*, "Sending and Receiving Messages," Setting and Overriding Target Connector Properties at Runtime.

Target Connector Passwords

You must encrypt all required and optional target connector passwords.

See Chapter 5, "Managing Integration Gateways," Encrypting Passwords, page 44.

Properties for HTTP URLs

The following connectors communicate over HTTP:

- PeopleSoft target connector.
- HTTP target connector.
- PeopleSoft 8.1 target connector.
- AS2 target connector.

For the HTTP target connector you can specify only one primary URL (PRIMARYURL) per node. The primary URL is the URL of the external system that handles the request.

However, you may specify more than one backup URL (BACKUPURL). Upon the failure of a transaction to the primary URL, the message is sent to any backup URLs one at a time. When a transaction that is sent to a URL succeeds, the other URLs are not used. If all URLs fail, the appropriate action and message is relayed to the calling module. The message and the node/URL failure is noted in the database or in the PeopleSoft Integration Broker Monitor.

Note. If the property ID is *HEADER*, then the target connector retrieves the information from a getHeader method call on the ConnectorInfo object, which resides on the IBRequest object. All other properties can be retrieved from a getFieldValue method call on the ConnectorInfo object.

Properties for Message Compression and Encoding

When the local integration gateway sends messages to a remote gateway, it ensures that they are compressed and base64 encoded. However, by default, when it sends messages directly to any node, it sends them uncompressed and unencoded. You can change this setting for transactions that use the following connectors:

- HTTP target connector.
- JMS target connector.
- FTP target connector.
- AS2 target connector.
- SMTP target connector.
- Simple file target connector.

Use the node-level SendUncompressed property for the appropriate connector. You can change the current value of this property specified for a given node by using the Connectors page of the node definition, or you can override the value for a single transaction by using the Connectors page of the node transaction detail. If you set the property's value to *No,* it sends messages compressed and base64 encoded.

See Chapter 7, "Adding and Configuring Nodes," page 127.

Note. If nonrepudiation is in effect for a message, the SendUncompressed property is not used, and the message is always sent compressed and base64 encoded.

Working With the PeopleSoft Connectors

This section provides an overview of the PeopleSoft connectors and discusses how to:

- Use the PeopleSoft listening connector.
- Use the PeopleSoft target connector.

Understanding the PeopleSoft Connectors

The PeopleSoft listening and target connectors establish the primary connection between a PeopleSoft application's integration engine and its designated local gateway. They also support secure HTTPS communications if SSL encryption is configured on the gateway machine.

Using the PeopleSoft Listening Connector

The PeopleSoft listening connector receives requests from integration participants in the PeopleSoft internal messaging format. Like the HTTP listening connector, the PeopleSoft listening connector is implemented as a Java HTTPServlet object. However, it receives requests in PeopleSoft Multipurpose Internet Mail Extensions (MIME) format. A PeopleSoft integration engine sends messages formatted in MIME over HTTP. The PeopleSoft listening connector receives these messages as POSTs (GET requests cannot be made in this way) and immediately converts the MIME input into a Java string object.

The PeopleSoft listening connector logs these requests and then invokes the gateway manager to unmarshall the string into an IBRequest object. The gateway manager invokes the target connector specified in the ConnectorClassName field in the IBRequest, which is derived from the node definition on the source integration engine. The gateway manager returns the responses to the connector, where they are logged and sent back to the original requesting systems, typically integration engines.

The URL for the PeopleSoft listening connector is http://*gatewayserver* /PSIGW/PeopleSoftListeningConnector, where *gatewayserver* is the machine name and port, host name, or IP address of the web server hosting the gateway.

Note. Third-party applications and PeopleSoft releases that don't include PeopleSoft Integration Broker should not send messages to this connector unless they can produce a properly MIME-encoded, PeopleSoft formatted message.

Using the PeopleSoft Target Connector

The PeopleSoft target connector initiates conversation with a PeopleSoft application's integration engine over a Oracle Jolt connection in the PeopleSoft internal messaging format. The integration gateway sends messages to a specific integration engine based on the destination node specified in an incoming message. Use this connector to send messages only to PeopleSoft applications that use PeopleSoft Integration Broker.

The connector ID for the PeopleSoft target connector is *PSFTTARGET*.

Gateway-Level Connector Properties

There are no gateway-level connector properties specific to this connector; however, it uses both the node-specific and default Oracle Jolt connect string properties in the integrationGateway.properties file to determine where to send the messages.

See Chapter 5, "Managing Integration Gateways," Setting General Connection Properties, page 47.

Node-Level Connector Properties

There are no node-level connector properties for the PeopleSoft target connector.

Working With the HTTP Connectors

This section provides an overview of the HTTP connectors and discusses how to:

- Use the HTTP listening connector.
- Use the HTTP target connector.
- Comply with message formatting and transmission requirements.
- Run the gateway behind a proxy server.

Understanding the HTTP Connectors

The HTTP listening and target connectors provide a web-standard method for an integration gateway to exchange messages with both PeopleSoft and third-party applications using the HTTP GET and POST methods. They also support secure HTTPS communications if SSL encryption is configured on the gateway machine.

Using the HTTP Listening Connector

The HTTP listening connector monitors specific ports for incoming HTTP messages. It's implemented as a Java HTTPServlet object. The URL for the HTTP listening connector is http://*gatewayserver* /PSIGW/HttpListeningConnector, where *gatewayserver* is the machine name and port, host name, or IP address of the web server hosting the gateway.

The HTTP listening connector accepts compressed and base 64-encoded data.

PeopleSoft HTTP Message Parameters

You must specify several required parameters in messages that you send to the HTTP listening connector. There are also several optional parameters.

These parameters, also known as *credentials,* are metadata specific to each message that the HTTP listening connector processes. These parameters supply authentication information and descriptive details about how the message is processed. For each message that you send to the connector, PeopleSoft Integration Broker uses the parameters that you supply to create an IBRequest that it uses to process and service the request internally. The following table describes the parameters:

Parameter	*Description*
OperationName	Specify the external alias name.
OperationType	(Optional.) Specify the type of message that is sent. Values are: • *Sync:* The message is synchronous. • *Async:* The message is asynchronous. • *Ping:* The message is used to ascertain whether the target node is active or inactive.
From	Specify the name of the node sending the request. This field is not required if you are invoking SSL encryption and addressing an HTTPS URL.
Password	Enter the password as it appears in the target node's definition for the source node. The target node authenticates the password when it receives the message. This parameter is required only if password authentication is enabled for the source node definition in the target database.
OrigUser	(Optional.) Specify the user ID from which the message was initially generated.
OrigNode	(Optional.) Specify the name of the node that started the process.
OrigProcess	(Optional.) Specify the name of the process on the source system that sent the message. For example, a message published from the Inventory Definitions page has a process name of INVENTORY DEFIN.
OrigTimeStamp	(Optional.) Specify the time at which the original request was created.
FinalDestination	(Optional.) Specify the name of the node that will ultimately receive the message. This is common when a PeopleSoft Integration Broker hub is used.

Parameter	*Description*
To	Specify the name of the node that will receive the message. This parameter is optional if you specified a default target node using the Default Application Server Jolt connect string properties in the integrationGateway.properties file. See Chapter 5, "Managing Integration Gateways," Setting General Connection Properties, page 47.
SubQueue	(Optional.) Specify the name of a partitioning subqueue to be created at runtime for the message. All messages with the same value for this parameter will be processed in the same subqueue. Unlike the subqueue created by selecting partitioning fields in a queue definition, the subqueue that you specify here has no qualifying criteria except the name that you enter. Field-based partitioning is ignored for messages with this parameter. See *Enterprise PeopleTools 8.50 PeopleBook: PeopleSoft Integration Broker*, "Managing Service Operation Queues," Applying Queue Partitioning.
NonRepudiation	(Optional.) Specify whether the message content in the request should be processed using nonrepudiation logic. Values are: • *Y:* Use nonrepudiation logic. • *N:* Don't use nonrepudiation logic.
MessageName	(Optional.) Specify the name of the message. This parameter is used for backward compatibility with previous PeopleTools releases.
MessageVersion	(Optional.) Specify which version of the message is sent. This parameter is used for backward compatibility with previous PeopleTools releases.
ExternalMessageID	(Optional.) Unique identifier for a message. The ID must not exceed 70 characters. See Using External Message IDs later in this section.

The PeopleSoft HTTP message parameters can be passed with inbound messages to the HTTP listening connector using several methods, and they are transmitted with outbound messages by the HTTP target connector.

See Chapter 6, "Using Listening Connectors and Target Connectors," Complying With Message Formatting and Transmission Requirements, page 77.

Using External Message IDs

You can specify an external message ID as a parameter in the HTTP listening connector to uniquely identify a message in PeopleSoft Integration Broker, thus ensuring that no duplicate messages are delivered to the system. The ExternalMessageID parameter is optional, but if you do specify this parameter, it must be unique and contain no more than 70 characters.

The HTTP listening connector can receive an external message ID in:

- Query strings.
- HTTP headers.
- SOAPAction headers.
- PeopleSoft IBRequest XML

The following example shows passing an external message ID in a query string:

```
http://localhost/PSIGW/HttpListeningConnector?From=QE_UNDERDOG&To=
QE_LOCAL&Operation=QE_SYNC_MSG.VERSION_1
ExternalMessageID=UniqueId0006
```

The following example shows passing an external message ID in an HTTP header:

```
ExternalMessageID: UniqueId0006
```

The following example shows passing an external message ID in a SOAPAction header:

```
http://peoplesoft.com/QE_SYNC_MSG/QE_UNDERDOG/password/QE_LOCAL/
UniqueId0006
```

The following example shows passing an external message ID in PeopleSoft IBRequest XML:

```
<?xml version="1.0"?>
<IBRequest>
    <From>
        <RequestingNode>QE_UNDERDOG</RequestingNode>
        <OrigTimeStamp>2003-09-29T00:37:30.790-0800</OrigTimeStamp>
        <ExternalMessageID>UniqueId0006</ExternalMessageID>
    /From>
    <ExternalOperationName>QE_SYNC_MSG.VERSION_1</ExternalOperationName>
    <OperationType>sync</OperationType>
    <To>
        <DestinationNode>QE_LOCAL</DestinationNode>
    </To>
    <ContentSections>
        <ContentSection>
            <Headers>
            <version>VERSION_1</version>
            </Headers>
            <Data><![CDATA[<?xml version="1.0"?><QE_SYNC_MSG/>]]></Data>
        </ContentSection>
    </ContentSections>
</IBRequest>
```

Using the HTTP Listening Connector to Receive Message Segments

The HTTP listening connector is segment-aware and you may use it to receive message segments from integration partners.

See *Enterprise PeopleTools 8.50 PeopleBook: PeopleSoft Integration Broker*, "Sending and Receiving Messages," Working With Message Segments.

Using the HTTP Target Connector

The HTTP target connector enables you to exchange messages with non-PeopleSoft systems using the HTTP protocol. The HTTP target connector uses SSL for all basic security services, including client-side authentication.

The HTTP target connector also supports the Simple Object Access Protocol (SOAP) XML format.

The connector ID for the HTTP target connector is *HTTPTARGET*.

IBInfo Data Contained in HTTP Headers

A message has two parts—the transaction data and the IBInfo header that is the routing envelope used by PeopleSoft Integration Broker. In the event that a receiving system wants to make use of the IBInfo data, IBInfo header information is included when publishing messages to non-PeopleSoft systems when using the HTTP target connector or the JMS target connector.

When using the HTTP target connector to send messages to non-PeopleSoft systems, the following IBInfo data is contained in the HTTP headers. The content of the message (message body) is not impacted.

- ExternalOperationName
- OperationType
- OrigTimeStamp
- NonRepudiation
- To
- From

Gateway-Level Connector Properties

The HTTP target connector provides the option of routing through proxy servers. To enable this capability, you must set the domain name of the proxy server and the port number of the proxy server in the integrationGateway.properties file:

See Chapter 6, "Using Listening Connectors and Target Connectors," Running Integration Gateways Behind Proxy Servers, page 85.

Node-Level Connector Properties

The HTTP target connector features properties that correspond to standard HTTP 1.1 header fields, as well as several custom properties that are documented in the following table. The World Wide Web Consortium (W3C) web site provides complete documentation for the standard header fields.

See http://www.w3.org/Protocols/rfc2616/rfc2616.html.

Property ID	*Property Name*	*Description*
HTTPPROPERTY	Method	Specify the HTTP method used to send messages. The valid values are: • *POST* (the default). • *GET.*
HTTPPROPERTY	SOAPUpContent	(Optional.) Automatically wrap outbound transactions in SOAP format. The valid values are: • *Y.* Outbound messages are wrapped in SOAP format. • *N.* Outbound message are not wrapped in SOAP format.
PRIMARYURL	URL	Specify the URL to which messages are sent using this connector.
BACKUPURL	URL	(Optional.) Specify the URL to which messages can be sent if the primary URL is inaccessible.
HEADER	SendUncompressed	Specify whether to send messages decompressed. Options are: • *Y:* Send messages decompressed and decoded. (Default.) • *N:* Send messages compressed and base64 encoded.
HEADER	Proxy-Authorization	Specify the user ID and password for proxy authentication. See Chapter 6, "Using Listening Connectors and Target Connectors," Running Integration Gateways Behind Proxy Servers, page 85.
HEADER	SOAPAction	(Optional.) Enable third-party systems (for example, Universal Description, Discovery, and Integration (UDDI) sites) to receive SOAP transactions over HTTP. The default value is "" (a null string).

Property ID	Property Name	Description
HEADER	TimeOut	Specify the time in milliseconds for the connector to wait for the message to transmit. If the timeout period expires without a successful transmission, the transaction fails. The default value is *50000* (50 seconds).

Using the Content-Type Property

One of the optional gateway-level properties you can set for the HTTP target connector is Content-Type.

When the HTTP target connector property Content-Type is *application/x-www-form-urlencoded*, the connector converts the content string to MIME format.

Encoding Strings

When encoding a string, the following rules apply:

- The alphanumeric characters "a" through "z", "A" through "Z" and "0" through "9" remain the same.
- The special characters ".", "-", "*", and "_" remain the same.
- The space character " " is converted into a plus sign "+".
- All other characters are unsafe and are first converted into one or more bytes. Then each byte is represented by the three-character string "%xy," where xy is the two-digit hexadecimal representation of the byte.

Using the HTTP Target Connector to Send Message Segments

The HTTP target connect is segment-aware and you may use it to send message segments to integration partners.

See *Enterprise PeopleTools 8.50 PeopleBook: PeopleSoft Integration Broker*, "Sending and Receiving Messages," Working With Message Segments.

Complying With Message Formatting and Transmission Requirements

This section discusses:

- The PeopleSoft XML message wrapper.
- The PeopleSoft non-XML message element.
- Passing HTTP parameters.
- Specifying message destinations in HTTP headers.
- Adding nonrepudiation signatures.

- Submitting cookies in the HTTP header.
- Responses to inbound request messages.
- Submitting SOAP messages.

This section directly addresses the issue of third parties that format and transmit messages to the HTTP listening connector; third parties should also expect the HTTP target connector to format and transmit outbound messages using the same standards.

The PeopleSoft XML Message Wrapper

At a minimum, when you submit message content to the HTTP listening connector, you submit it—preceded by the following XML version declaration—inside a simple XML wrapper:

```
<?xml version="1.0"?>
  <![CDATA[your_message_content]]>
```

Upon receiving the message, the integration gateway strips off the outer elements, leaving the message content with its original XML version declaration to be handled by PeopleSoft Integration Broker:

```
<?xml version="1.0"?>your_message_content
```

The message content can comply with the PeopleSoft rowset-based message format, which you can manipulate using the PeopleCode Rowset class. It can also be nonrowset-based XML-DOM-compliant data, which you can manipulate with nonrowset PeopleCode. Both formats are compatible with Application Engine transform programs, in which you can manipulate the message content using both PeopleCode and Extensible Stylesheet Language Transformation (XSLT) code.

The following template shows how a message in PeopleSoft rowset-based message format fits into the XML wrapper (data omitted for readability):

```
<?xml version="1.0"?>
     <![CDATA[<?xml version="1.0"?>
      <psft_message_name>
         <FieldTypes>
            ...
         </FieldTypes>
         <MsgData>
            ...
         </MsgData>
      </psft_message_name>]]>
```

Note. *Psft_message_name* is the name of the message definition in the PeopleSoft database.

The PeopleSoft Non-XML Message Element

If you're submitting a non-XML message, you must insert the message content into a special element containing its own CDATA tag, as follows:

```
<?xml version="1.0"?>
   <![CDATA[<?xml version="1.0"?>
      <any_tag psnonxml="yes">
         <![CDATA[your_nonXML_message_content]]>
      </any_tag>]]>
```

Note. *Any_tag* can be any tag that you want to use. This is an XML-DOM-compliant method of transmitting non-XML data.

The following restrictions apply to the content of non-XML messages, such as those in comma-separated value (CSV) or PDF format:

- If the message content is non-XML text, it must be encoded as characters that are compliant with Unicode Transformation Format 8 (UTF-8).
- If the message content is non-text (binary), it must be encoded in base64 format.

Upon receiving the message, the integration gateway strips off the outer elements, leaving the non-XML message content inside a valid XML-DOM-compliant wrapper with its original XML version declaration.

Passing HTTP Parameters

You can pass parameters to the HTTP listening connector in:

- The PeopleSoft message wrapper, through an HTTP POST.
- The HTTP header, through an HTTP GET or POST.
- The URL query string, through an HTTP GET or POST.

The only HTTP parameters that you must provide for basic messaging are MessageName and RequestingNode. If you pass them in the PeopleSoft message wrapper, you must embed them in an XML structure along with the CDATA element containing the message content. Following is the minimum wrapper structure required to pass the parameters this way:

```
<?xml version="1.0"?>
<IBRequest>
   <ExternalOperationName>psft_operation_name</ExternalOperation
    Name>
   <From>
      <RequestingNode>psft_node_name</RequestingNode>
   </From>
   <ContentSections>
      <ContentSection>
         <Data>
            <![CDATA[<?xml version="1.0"?>your_message_content]]>
         </Data>
      </ContentSection>
   </ContentSections>
</IBRequest>
```

Note. *Psft_message_name* and *psft_node_name* are the names of the message definition and the sending system's node definition in the PeopleSoft database.

If you want to pass all of the HTTP message parameters in the PeopleSoft message wrapper, you embed them in the XML wrapper structure as follows (required parameters are shown emphasized, and element values are omitted for readability):


```
<?xml version="1.0"?>
<IBRequest>
   <ExternalOperationName/>
   <OperationType/>
   <From>
      <RequestingNode/>
      <Password/>
      <OrigUser/>
      <OrigNode/>
      <OrigProcess/>
      <OrigTimeStamp/>
   </From>
   <To>
      <FinalDestination/>
      <DestinationNode/>
      <SubChannel/>
   </To>
   <ContentSections>
      <ContentSection>
         <NonRepudiation/>
         <MessageVersion/>
         <Data>
            <![CDATA[<?xml version="1.0"?>your_message_content]]>
         </Data>
      </ContentSection>
   </ContentSections>
</IBRequest>
```

The following template shows the format for passing HTTP message parameters in the HTTP message header. The optional parameters can be omitted if not needed. The HTTP header format is as follows (required parameters are shown emphasized):

```
OperationName: OperationName
OperationType: sync|async|ping
From: RequestingNode
Password: Password
OrigUser: OrigUser
OrigNode: OrigNode
OrigProcess: OrigProcess
OrigTimeStamp: OrigTimeStamp
FinalDestination: FinalDestination
To: DestinationNode
SubQueue:  SubQueue
NonRepudiation: Y|N
```

Warning! Whether you send message parameters in the message wrapper or in the HTTP header, those parameters—including the password—aren't secure if you don't encrypt the message. You can secure messages by implementing SSL encryption.

The following template shows the format for passing HTTP message parameters in a URL query string. Include all of the parameter variables, even if you don't supply values for some of them. With only the required parameters, the URL query string looks like the following (required parameters are emphasized):

```
http://gatewayserver/PSIGW/HttpListeningConnector?&Operation=Operation
Name&OperationType=&From=RequestingNode&Password=&OrigUser=&OrigNode=
&OrigProcess=&OrigTimeStamp=&FinalDestination=&To=&SubQueue=
&NonRepudiation=&MessageVersion=
```

The full URL query string format is:

```
http://gatewayserver/PSIGW/HttpListeningConnector?&Operation=Operation
Name&OperationType=[sync|async|ping]&From=RequestingNode&Password=
Password&OrigUser=OrigUser&OrigNode=OrigNode&OrigProcess=OrigProcess&
OrigTimeStamp=OrigTimeStamp&FinalDestination=FinalDestination&To=DestinationNode⇒
&SubQueue=SubQueue&NonRepudiation=[Y|N]&MessageVersion=MessageVersion
```

Warning! URL query strings are always transmitted in clear text, so your parameters are visible to the world. This means that using a query string to send message parameters—such as a password—is highly insecure. Consequently, it is not recommended.

Using an HTTP POST is the only way that you can send message content to PeopleSoft Integration Broker through the HTTP listening connector. However, you can use an HTTP GET when you don't need to post message content. In this case, you pass the HTTP connector properties in the URL query string or in the HTTP header, but you don't insert any message content or XML wrapper. For example, you might have requests for information (queries), such as a request for a customer list. In this case, you need to specify only the message name (for example, CUSTOMER_LIST_REQUEST) and the name of the requesting node in the URL query string or the HTTP header.

Specifying Message Destinations in HTTP Headers

When message credentials are supplied in HTTP headers, the "To:" (destination node) specification is ignored. PeopleSoft Integration Broker uses the Default Application Server node entry in the integrationGateway.properties file as the destination node, not the "To:" entry from the headers. If no default application server entry is specified in the integrationGateway.properties file, the follow error is generated:

```
<?xml version="1.0"?>
<IBResponse type="error">
<DefaultTitle>Integration Broker Response</DefaultTitle>
<StatusCode>20</StatusCode>
<MessageID>10201</MessageID>
<DefaultMessage>null</DefaultMessage>
</IBResponse>
```

You can specify destination node information in the SOAPAction field or HTTP query string.

Note. If using SOAP, PeopleSoft Integration Broker takes all IBInfo from the SOAPAction field, not from the HTTP header or HTTP query string.

Adding Nonrepudiation Signatures

If you're working with a nonrepudiated message, its signature must be located at the same level as the message data.-The message doesn't need to be formatted with the PeopleSoft rowset hierarchy, as long as it's enclosed in valid XML and has the signature section as specified by the W3C. The following template describes a nonrepudiation signature alongside the PeopleSoft rowset-based format message it represents, within the ContentSection element of the PeopleSoft XML message wrapper (the tags you must add for nonrepudiation are in bold):

```
<ContentSections>
   <ContentSection>
      <NonRepudiation>Y</NonRepudiation>
          <Data>
        <?xml version="1.0"?>
         <any_tag>
            <data>
               <![CDATA[<?xml version="1.0"?>your_message_content]]>
            </data>
            <Signature>
               <SignedInfo>
                <CanonicalizationMethod/>
                <SignatureMethod/>
                <Reference>
                   <DigestMethod>
                      <DigestValue>
                      ...
                      </DigestValue>
                   </DigestMethod>
                </Reference>
               </SignedInfo>
             <SignatureValue>
             ...
             </SignatureValue>
          </Signature>
        </any_tag>
       </Data>
   </ContentSection>
</ContentSections>
```

Note. *Any_tag* can be any tag that you want to use, such as *My_NR_Message.*

You can find more information about the proposed standard for XML signature syntax and processing at the W3C web site.

See http://www.w3.org/TR/xmldsig-core/.

Important! In PeopleSoft Integration Broker, all signatures use line feeds for newlines, so the nonrepudiation signature cannot include any carriage return and line feed (CR/LF) pairs. A non-PeopleSoft application must strip out the carriage returns before inserting the signature and sending the message.

Note. To handle nonrepudiated messages, you must install node-based digital certificates on the sending and receiving systems and configure the message and channel definitions to use the nonrepudiation feature.

See Chapter 11, "Setting Up Secure Integration Environments," Implementing Nonrepudiation, page 217.

Submitting Cookies HTTP Headers

The HTTP listening connector supports cookies. Cookies that are passed as part of a message request to the HTTP listening connector are processed, read, and manipulated by the receiving PeopleCode in the application. You enter cookies in the HTTP message header. For example:

```
Cookie: favoritecolor=green; path=/; expires Mon, 10-Dec-2007 13:46:00 GMT
```

In this example, the header entry would result in a cookie named favoritecolor. The value of favoritecolor is *green.* This cookie has a path of `/`, meaning that it is valid for the entire site, and it has an expiration date of December 10, 2007 at 1:46 p.m. Greenwich Mean Time (GMT).

See *Enterprise PeopleTools 8.50 PeopleBook: PeopleSoft Integration Broker*, "Sending and Receiving Messages," Handling Cookies.

Responses to Inbound Requests

PeopleSoft Integration Broker responds to inbound requests in one of three ways:

- For a successfully received *synchronous* transmission, the integration gateway passes the request to the integration engine.

  The integration engine generates and passes back through the listening connector a response in a format determined by the applicable node, service operation definition and routing definition for the request.

- For a successfully received *asynchronous* transmission, the integration gateway immediately returns a simple XML acknowledgment message.

  The following example shows a successful asynchronous acknowledgment:

```
<?xml version="1.0"?>
<IBResponse type="success">
   <DefaultTitle>Integration Broker Response</DefaultTitle>
   <StatusCode>0</StatusCode>
   <TransactionID>UNDERDOG.QE_SALES_ORDER_ASYNC_CHNL.20</TransactionID>
</IBResponse>
```

- For an *unsuccessful* transmission, the integration gateway immediately returns a simple XML error message in a standard XML error format for all requests (except SOAP requests), if error handling is invoked in the integration gateway.

  The following is an example of this standard error response:

```
<?xml version="1.0"?>
<IBResponse type="error">
   <DefaultTitle>Integration Broker Response</DefaultTitle>
   <StatusCode/>
   <MessageID/>
   <DefaultMessage/>
   <MessageParameters>
      <Parameter/>
   </MessageParameters>
</IBResponse>
```

Submitting SOAP Messages

SOAP messages support a subset of the HTTP message parameters— two required parameters and two optional parameters. You pass them to the HTTP listening connector in a SOAP-specific HTTP header. Concatenate them in a string, with each parameter preceded by a forward slash (/). They must appear in the following order:

```
http://peoplesoft.com/OperationName/RequestingNode/Password/DestinationNode
```

The following example shows where the parameter string belongs in a SOAP HTTP header:

```
POST /get_BindingDetail HTTP/1.1
Host: www.someOperator.com
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
SOAPAction: http://peoplesoft.com/PURCHASE_ORDER/MY_NODE/
PSFT_PASS/PSFT_NODE
```

Note. The SOAPAction must always be in the HTTP header, not contained within the IBRequest XML.

Because the last two parameters are optional, you can exclude them; however, you must still include the forward slashes. This example excludes the password:

```
SOAPAction: http://peoplesoft.com/PURCHASE_ORDER/MY_NODE//PSFT_NODE
```

Note. The SOAPAction format for previous PeopleTools releases is still supported. This format had the parameters concatenated in a string separated by pound signs ("#"): SOAPAction: #PURCHASE_ORDER#MY_NODE#PSFT_PASS#PSFT_NODE

Warning! When you send message parameters in the SOAP header, those parameters—including the password—aren't secure if you don't encrypt the message. You can secure messages by implementing SSL encryption.

If an error occurs on the integration gateway during processing, a SOAP-specific XML error is generated instead of a standard XML error. Following is an example of an error in SOAP-specific XML format:

```
<?xml version="1.0"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
   <SOAP-ENV:Body>
      <SOAP-ENV:Fault>
         <faultcode>SOAP-ENV:Server</faultcode>
         <faultstring>Server Error</faultstring>
         <detail>
            <IBResponse type="error">
               <DefaultTitle>Integration Broker Response</DefaultTitle>
               <StatusCode>10</StatusCode>
               <MessageID>10731</MessageID>
               <DefaultMessage></DefaultMessage>
            </IBResponse>
         </detail>
      </SOAP-ENV:Fault>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Understanding HTTP Status Codes

This section describes HTTP status codes for non-SOAP and SOAP messages.

Status Codes for Non-SOAP Messages

The following list summarizes HTTP status codes for non-SOAP messages:

- For an asynchronous message, HTTP status codes *200* to *299* indicate a message status of Success.
- For a synchronous message, the HTTP status code *200* indicates a message status of Success.
- HTTP status code *404* indicates that the server has not found anything matching the Request-URI. In this case, an ExternalSystemContactException is generated on the integration gateway and the message status goes to Retry.

- HTTP status code *503* indicates that the server is currently unable to handle the request due to temporary server overload or maintenance. In this case, an ExternalSystemContactException is generated on the integration gateway and the message status goes to Retry.
- All other HTTP status codes generate an ExternalApplicationException. The status of these messages goes to Error.

Status Codes for SOAP Messages

This section summarizes HTTP status codes for SOAP messages.

If you are following SOAP 1.1 standards, the HTTP status code 500 indicates an Error.

If you are following SOAP 1.2 standards, the following HTTP status codes apply:

- HTTP status code 400 can mean any of the following:
  - InvalidMessageException
  - MessageMarshallingException
  - MessageUnmarshallingException
  - Malformed HTTP/XML
- HTTP status code 405 indicates that the method is not POST.
- HTTP status code 415 indicates the content type is not text/xml.
- HTTP status code 500 can mean any of the following:
  - ExternalSystemContactException
  - ExternalApplicationException
  - GeneralFrameworkException

Running Integration Gateways Behind Proxy Servers

When a proxy server is set up for a network on which the integration gateway resides, all HTTP transactions are routed through that proxy server automatically. The HTTP transport layer uses proxy server settings that you specify in the integrationGateway.properties file. The message is routed to the proxy server and then on to the internet. Only the HTTP target connector can use a proxy server.

Inserted in the HTTP message header of each transaction is a *Proxy-Authorization* header field containing a user ID and password. The proxy server uses these values to authenticate the message and then passes it on to its target.

Setting Proxy Web Server Properties

To run the integration gateway behind a proxy server:

1. Set the gateway-level properties.

   Uncomment and add values for the properties in the integrationGateway.properties file section labeled *Proxy webserver section.*

Property	*Description*
ig.proxyHost	Enter the domain name of the proxy web server; for example: `proxy.peoplesoft.com`
ig.proxyPort	Enter the port number of the proxy web server; for example: `80`

   The HTTP target connector reads these two properties and calls the setProxy function. In an outbound transaction, the request is redirected to the proxy server and the proxy server forwards the request to the destination URL.

2. Set the node-level property.

   You set the user ID and password required by the proxy server in the *HEADER, Proxy-Authorization* connector property. The integration gateway encodes the values that you provide, adds the required formatting, and sends it. The format is:

   `userid``:`*`password`*

See Also

Chapter 5, "Managing Integration Gateways," Using the integrationGateway.properties File, page 41

Working With the PeopleSoft Services Listening Connector

This section discusses how to:

- Set parameters for the PeopleSoft services listening connector.
- Pass parameters for the PeopleSoft services listening connector.
- Pass parameters to get XML schema, WSDL, and WSIL.

Understanding the PeopleSoft Services Listening Connector

The PeopleSoft services listening connector is used for inbound integrations with web services.

SOAP Messages

If the inbound request is a SOAP message:

- The SOAPAction must take the following format:

  ```
  SOAPActon:<External_alias_name>
  ```

- The response message should also be in SOAP format. If it is not, it should be wrapped in SOAP format.
- Any errors generated are in SOAP format or wrapped in the SOAP fault tag and returned to the sender.

Setting Parameters for the PeopleSoft Services Listening Connector

The same required and optional parameters that you can set for the HTTP listening connector pertain to the PeopleSoft services listening connector. For a list of the required and optional parameters, see the Using the HTTP Listening Connector section presented previously in this chapter.

See Chapter 6, "Using Listening Connectors and Target Connectors," Using the HTTP Listening Connector, page 71.

Passing Parameters to the PeopleSoft Services Listening Connector

This section discusses how to pass parameters to the PeopleSoft services listening connector.

Passing Parameters to the PeopleSoft Services Listening Connector in URL Query Format

You can pass parameters to the PeopleSoft service listening connector using a URL query string using the following format:

```
http://<machinename>:<port>/PSIGW/PeopleSoftServiceListening
Connector?Operation=OperationName
```

The following format is also supported:

```
http://<machinename>:<port>/PSIGW/PeopleSoftServiceListening
Connector?Operation=<OperationName>>&To=<ReceiverNode>&From=
<SenderNode>&OperationType=<Type>
```

Passing Parameters to the PeopleSoft Services Listening Connector in Path Format

You can pass parameters to the PeopleSoft service listening connector using a path format using the following format:

```
http://127.0.0.1/PSIGW/PeopleSoftServiceListeningConnector/
SERVICE_OPERATION.VERSION.xsd
```

Passing Parameters to Get XML Schema, WSDL and WSIL

You can use query format or path format to get XML schema, WSDL and WSIL.

Using Query Format to Get XML Schema, WSDL and WSIL

Use the following query format to get XML schema:


```
http://<machinename>:<port>/PSIGW/PeopleSoftServiceListeningConnector?Operation=
GetSchema&xsd=SERVICE_OPERATION.VERSION
```

Use the following query format to get WSDL:

```
http://<machinename>:<port>/PSIGW/PeopleSoftServiceListeningConnector?Operation=
GetWSDL&wsdl=SERVICE_OPERATION.VERSION
```

Use the following query format to get WSIL:

```
http://<machinename>:<port>/PSIGW/PeopleSoftServiceListeningConnector?Operation=
GetWSIL
```

Using Path Format to Get XML Schema, WSDL and WSIL

Use the following path format to get XML schema:

```
http://<machinename>:<port>/PSIGW/PeopleSoftServiceListeningConnector/
<REMOTENODE>/<OperationName>.<version>.xsd
```

Use the following path format to get WSDL:

```
http://<machinename>:<port>/PSIGW/PeopleSoftServiceListeningConnector/
<REMOTENODE>/<OperationName>.<version>.wsdl
```

Use the following path format to get WSIL:

```
http://<machinename>:<port>/PSIGW/PeopleSoftServiceListeningConnector/
<REMOTENODE>/inspection.wsil
```

Working With the PeopleSoft 8.1 Connectors

This section provides an overview of the PeopleSoft 8.1 connectors and discusses how to:

- Use the PeopleSoft 8.1 listening connector.
- Use the PeopleSoft 8.1 target connector.

Understanding the PeopleSoft 8.1 Connectors

The PeopleSoft 8.1 listening and target connectors enable communication between PeopleSoft 8.1x applications and an integration gateway using PeopleSoft Application Messaging technology. To the PeopleSoft 8.1x application, the gateway appears to be another PeopleSoft 8.1x application, so no change in the messaging development process is needed. The connectors also support secure HTTPS communications if SSL encryption is configured on the gateway machine.

Note. The PeopleSoft 8.1 connectors are intended for use for integrations with PeopleSoft systems running PeopleTools 8.1x.

See Also

Chapter 11, "Setting Up Secure Integration Environments," Implementing Web Server SSL Encryption, page 188

Using the PeopleSoft 8.1 Listening Connector

In PeopleSoft 8.1x systems, PeopleSoft Application Messaging generates highly structured XML messages that are designed to be sent to PeopleSoft 8.1x Application Messaging gateways. The PeopleSoft 8.1 listening connector mimics the role of the Application Messaging gateway by transparently receiving and processing PeopleSoft 8.1x messages. This connector transforms inbound PeopleSoft 8.1x messages into PeopleSoft Integration Broker formatted XML messages that can be processed by the integration gateway and ultimately by the integration engine. This conversion is necessary because the two message formats are distinctly different.

The URL for the PeopleSoft 8.1 listening connector is http://*gatewayserver*/PSIGW/PS81ListeningConnector, where *gatewayserver* is the machine name and port, host name, or IP address of the web server hosting the gateway.

This connector automatically handles base64–encoded and compressed messages, as well as uncompressed messages.

Using the PeopleSoft 8.1 Target Connector

This connector enables the gateway to communicate with PeopleSoft 8.1x applications that use PeopleSoft Application Messaging technology. It converts outbound messages to the Application Messaging native format. Messages from the PeopleSoft Integration Broker system reach the PeopleSoft 8.1x system through the Application Messaging gateway on the PeopleSoft 8.1x system.

The PeopleSoft 8.1 target connector uses the HTTP target connector to manage the HTTP communication with the PeopleSoft 8.1x Application Messaging gateway. The PeopleSoft 8.1 target connector focuses on messaging semantics, instead of communication details; it constructs an Application Messaging XML document and sends it using the HTTP target connector. The PeopleSoft 8.1 target connector detects the status of returned responses by the value in the ReturnCode field in the XML response.

The connector ID for the PeopleSoft 8.1 target connector is *PSFT81TARGET*.

Gateway-Level Connector Properties

The PeopleSoft 8.1 target connector has one gateway-level property, in the section of the integrationGateway.properties file labeled *DELIVERED CONNECTOR CONFIGURATION Section.* This property specifies where the connector can send messages if a target URL isn't specified in the connector's node-level properties. Specify the URL as follows:

`ig.connector.amtargetconnector.url=`*`peoplesoft_8.1x_application_messaging_gateway`*

You can override this value by specifying a different URL in the node-level connector properties, in the node definition for the PeopleSoft 8.1x target node, or in the transaction definition for the message.

Node-Level Connector Properties

The following table describes the node-level connector properties:

Property ID	Property Name	Description
PSFT81TARGET	URL	Specify the PeopleSoft 8.1x Application Messaging gateway URL to which messages are sent using this connector.
HEADER	TimeOut	Specify the time in milliseconds for the connector to wait for the message to transmit. If the timeout period expires without a successful transmission, the transaction fails. The default value is *50000* (50 seconds).

Working With the JMS Connectors

This section provides an overview of the JMS connectors and discusses how to:

- Specify JNDIFactory class names.
- Use the JMS listening connector.
- Use the JMS target connector.

Understanding the JMS Connectors

The JMS listening and target connectors enable communication between JMS provider systems and an integration gateway using standard JMS protocols. PeopleSoft currently supports Java Native Directory Interface (JNDI) only for File System Context [fscontext] and RMI lookup.

Note. Check MyOracle Support for the JMS specification currently supported by PeopleTools. PeopleSoft Integration Broker's JMS listening connector and JMS listening connector are compliant with the specification version listed.

Supported JMS Providers

To use the JMS connectors, you must add specific Java archive (JAR) files to the Java CLASSPATH. The JAR files that you add to the CLASSPATH depend on the JMS provider with which you're communicating. The following JMS providers are supported:

JMS Provider	Required Files
Oracle WebLogic	N/A **Note.** PeopleSoft Integration Broker currently doesn't support using the IBM WebSphere web server with a WebLogic JMS provider.

JMS Provider	*Required Files*
IBM MQ Series	jms.jar, jndi.jar, fscontext.jar, com.ibm.mqjms.jar

Note. Not only can a gateway running on a Oracle WebLogic web server communicate with a WebLogic JMS provider, but both services can run on a single installation of WebLogic. However, the gateway still treats the JMS provider as a separate system, and it must be configured the same way as in any other scenario.

You can also add generic JMS providers for use with PeopleSoft Integration Broker.

See Chapter 6, "Using Listening Connectors and Target Connectors," Adding Generic JMS Providers, page 104.

Specifying JNDIFactory Class Names

You must set up the JNDIFactory class names for the JMS provider in the section of the integrationGateway.properties file labeled *JMS configuration Section.*

When you set the JMSProvider property, the provider name that you enter must match the provider in the JNDIFactory class name exactly. You must set this property for both the JMS listening connector and the JMS target connector. This property is case-sensitive.

JMS Provider	*Property*	*Description*
Oracle WebLogic	ig.jms.JMSProvider.JNDIFactory.Weblogic	Specify the JNDIFactory class name for a Oracle WebLogic JMS provider. The default value is: *weblogic.jndi.WLInitialContextFactory*
IBM MQ Series	ig.jms.JMSProvider.JNDIFactory. MQSeries	Specify the JNDIFactory class for an MQSeries JMS provider. The default value is: *com.sun.jndi.fscontext.RefFSContextFactory*

You can also specify a service provider that is not listed. For example, if you are using MSMQ, enter the following value for the property:

```
ig.jms.JMSProvider.JNDIFactory.MSMQ=com.sun.jndi.fscontext.RefFSContextFactory
```

Using the JMS Listening Connector

The JMS listening connector has two components: a subscriber and a queue listener. The JMS subscriber subscribes to different topics and the JMS queue listens on queues for new messages.

Note. The JMS listening connector always expects JMS messages in text format.

Receiving Messages

The JMS listening connector retrieves topics and queues that you have defined in integrationGateway.properties file. For each topic it starts a topic subscriber, and for each queue it starts a queue listener. When a message arrives either for a queue or topic, the JMS listening connector sends the message to the integration engine.

A parameter called ExternalMessageID is used to ensure that messages are received only once. When the JMS listening connector receives a message, it sets an external message ID in IBInfo and sends this information to the PeopleSoft Integration Broker with the message content. If the external message ID exists in IBInfo, the application server checks for duplicate messages. If a duplicate is found, an error is generated. The external message ID is optional. If specified, it must be unique and not exceed 70 characters.

Securing Messages to JMS Queues

PeopleSoft Integration Broker does not perform security validation checks on messages transmitted to JMS queues.

Note. JMS administrators must set up secure queues on providing systems.

Error Handling

If an error occurs during message processing, the JMS listening connector publishes the message back to either an error topic or an error queue. All error messages feature a header called ErrorDescription which contains a description of the error.

Note. If the application server returns the status *20*, the message is published to the error topic and the response is logged in the integration gateway message log.

To capture errors you must set error topic or error queue properties in the JMS Configuration Section of the integrationGateway.properties file. These properties are discussed later in this section. If both an error topic and an error queue are set up and configured, only the error queue will capture error messages.

JMS Queue Listener Properties

You can configure multiple queues in the section of the integrationGateway.properties file labeled *JMS Configuration Section.* To configure multiple queues, use the convention, *ig.queue1,ig.queue2,ig.queue3*, and so on.

Property	*Description*
ig.jms.Queues	Specify the number of queue listeners to instantiate.
ig.jms.Queue1	Specify the queue name.
ig.jms.Queue1.Provider	Specify the queue provider name.

Property	*Description*
ig.jms.Queue1.JMSFactory	Specify the JMSFactory name that is bound to JNDI for the queue.
ig.jms.Queue1.MessageSelector	(Optional.) Specify the message filter.
ig.jms.Queue1.URL	Specify the JMS provider's URL to JNDI.
ig.jms.Queue1.User	(Optional.) Specify the JMS queue user name.
ig.jms.Queue1.Password	(Optional.) Specify the JMS queue password. If you choose to specify a password, you must encrypt it. See Chapter 5, "Managing Integration Gateways," Encrypting Passwords, page 44.
ig.jms.Queue1.MessageName	This is a deprecated property and is being maintained for backwards compatibility only.
ig.jms.Queue1.MessageVersion	This is a deprecated property and is being maintained for backwards compatibility only.
ig.jms.Queue1.OperationName	(Optional.) Specify the name of the service operation and the service operation version. The format is: *ig.jms.Queue1.OperationName=OperationName.OperationVersion.*
ig.jms.Queue1.RequestingNode	(Optional.) Specify the name of the requesting node.
ig.jms.Queue1.DestinationNode	(Optional.) Specify the name of the destination node.
ig.jms.Queue1.NodePassword	(Optional.) Specify the password for the requesting node. If you choose to specify a password, you must encrypt it. See Chapter 5, "Managing Integration Gateways," Encrypting Passwords, page 44.
ig.jms.Queue1.SubChannel	(Optional.) Specify the name of the subchannel. Messages published to this queue go to the subchannel indicated.

JMS Topic Subscriber Properties

You can configure multiple topics, in the section of the integrationGateway.properties file labeled *JMS configuration Section.* To configure multiple topics, use the convention *ig.topic1,ig.topic2,ig.topic3,* and so on.

Property	*Description*
ig.jms.Topics	Specify the number of topic subscribers to instantiate.
ig. jms.Topic1	Specify the topic name.
ig. jms.Topic1.Provider	Specify the topic provider name.
ig. jms.Topic1.JMSFactory	Specify the JMSFactory name that is bound to JNDI for the topic.
ig. jms.Topic1.MessageSelector	(Optional.) Specify the message filter.
ig. jms.Topic1.URL	Specify the JMS provider's URL to JNDI.
ig. jms.Topic1.User	(Optional.) Specify the JMS topic user name.
ig. jms.Topic1.Password	(Optional.) Specify the JMS topic password. If you choose to specify a password, you must encrypt it. See Chapter 5, "Managing Integration Gateways," Encrypting Passwords, page 44.
ig.jms.Topic1.MessageName	This is a deprecated property and is being maintained for backwards compatibility only.
ig.jms.Topic1.MessageVersion	This is a deprecated property and is being maintained for backwards compatibility only.
ig.jms.Topic1.OperationName	(Optional.) Specify the name of the service operation and the service operation version. The format is: *ig.jms.Queue1.OperationName=OperationName.OperationVersion.*
ig.jms.Topic1.RequestingNode	(Optional.) Specify the name of the requesting node.
ig.jms.Topic1.DestinationNode	(Optional.) Specify the name of the destination node.
ig.jms.Topic1.NodePassword	(Optional.) Specify the password for the requesting node. If you choose to specify a password, you must encrypt it. See Chapter 5, "Managing Integration Gateways," Encrypting Passwords, page 44.

Property	Description
ig.jms.Topic1.SubChannel	(Optional.) Specify the name of the subchannel. Messages published to this topic go to the subchannel indicated.

Error Queue Properties

To capture JMS listening connector errors in an error queue, set the following properties in the *JMS Configuration Section* of the integrationGateway.properties file.

Property	Description
ig.jms.ErrorQueue	Specify the name of queue to which error messages are published.
ig.jms.ErrorQueue-Provider	Specify the name of the JMS provider.
ig.jms.ErrorQueue-User	(Optional.) Specify the JMS error queue user name.
ig.jms.ErrorQueue-Password	(Optional.) Specify the JMS error queue password. If you choose to specify a password, you must encrypt it. See Chapter 5, "Managing Integration Gateways," Encrypting Passwords, page 44.
ig.jms.ErrorQueue-JMSFactory	Specify the queue connection factory name.
ig.jms.ErrorQueue-Url	Specify the JMS provider's URL to JNDI.

Error Topic Properties

To capture JMS listening connector errors in an error topic, set the following properties in the *JMS Configuration Section* of the integrationGateway.properties file.

Property	Description
ig.jms.ErrorTopic	Specify the name of topic to which error messages are published.
ig.jms.ErrorTopic-Provider	Specify the name of the JMS provider.
ig.jms.ErrorTopic-User	(Optional.) Specify the JMS error topic user name.

Property	Description
ig.jms.ErrorTopic-Password	(Optional.) Specify the JMS error topic password. If you choose to specify a password, you must encrypt it. See Chapter 5, "Managing Integration Gateways," Encrypting Passwords, page 44.
ig.jms.ErrorTopic-JMSFactory	Specify the JNDIFactory name.
ig.jms.ErrorTopic-Url	Specify the JMS provider's URL to JNDI.

JMS Message Header Properties

For the JMS listening connector to process messages, you must set the following properties. You can set these properties in JMS message headers, the integrationGateway.properties file or in the body of the XML message.

You can specify JMS headers in the integrationGateway.properties file for both queues and topics. However you must be using separate queues or topics per requesting node/message combination.

You must supply the properties listed in the following table in the JMS message header when you publish messages from a JMS provider system to the integration gateway.

Property	Description
MessageName	Specify the name of service operation.
RequestingNode	Specify the requesting node name.
FinalDestinationNode	Specify the final destination nodes. If there are no values, set this property to *Null.*
DestinationNode	Specify the destination node names, separated with commas. If there are no values, set to "" (empty string).
NodePassword	Enter the node password. This password must be encrypted. See Chapter 5, "Managing Integration Gateways," Encrypting Passwords, page 44.

Property	Description
SubChannel	(Optional.) Specify the name of a partitioning subqueue to be created for the service operation at runtime. All service operations with the same value for this parameter are processed in the same subqueue. Unlike the subqueues created by selecting partitioning fields, the subqueue that you specify here has no qualifying criteria except the name that you enter. Field-based partitioning is not used for service operations with this parameter. See *Enterprise PeopleTools 8.50 PeopleBook: PeopleSoft Integration Broker*, "Managing Service Operation Queues," Applying Queue Partitioning.

The following example shows specifying JMS header properties in the body of an XML message.

```
<?xml version="1.0" ?>
   <IBRequest>
      <ExternalOperationName>JMS_MessageName</ExternalOperationName>
      <OperationType>Async_or_Synch</OperationType>
         <From>
            <RequestingNode>JMS_RequestingNode</RequestingNode>
            <Password>JMS_NodePassword</Password>
            <OrigUser></OrigUser>
            <OrigNode></OrigNode>
            <OrigProcess></OrigProcess>
            <OrigTimeStamp></OrigTimeStamp >
         </From>
         <To>
            <FinalDestination>JMS_FinalDestination</FinalDestination>
            <DestinationNode>JMS_DestinationNode</DestinationNode>
         </To>
         <ContentSections>
            <ContentSection>
               <NonRepudiation></NonRepudiation>
                <Data></Data>
            </ContentSection>
        </ContentSections>
   </IBRequest>
```

When the message received specifies synchronous mode, a reference to the temporary queue or topic must be set in the JMS message header for the JMS listening connector to determination the destination of the message response. The JMS listening connector also sets the JMS correlation ID when it sends the response so the requestor can properly associate the response with its corresponding request.

If any of the message header properties are missing, an error is logged and an error is published to an error topic or error queue. The message that the connector publishes to the error topic has a property call error and is set to *True*. The error message that is published contains the following information: default message, message ID, message set, message parameters, and body of the message sent.

Starting the JMS Listening Connector

You can start the JMS listening connector using the JMSListeningConnectorAdministrator servlet, or you can start it manually.

To start the JMS listening connector using the servlet:

1. Deploy the servlet under the web application PSIGW.
2. To start the servlet on start up of the web server, set the variable Load On Startup.

   When you set the Load On Startup option, the JMS listening connector automatically starts when you start the web server. Refer to the web server documentation for more details about starting the servlet automatically when the web server starts.

To start the JMS listening connector manually:

1. Open a browser and enter the following URL:

   http://localhost:port/PSIGW/JMSListeningConnectorAdministrator?Activity=Start
2. Press Enter.

The message 'JMS Listening Connector Started' displays.

If you experience problems starting the JMS listening connector, check the integration gateway error log file, errorlog.html, for additional information.

Shutting Down the JMS Listening Connector

You can shut down the JMS listening connector by stopping the JMSListeningConnectorAdministrator servlet.

To shut down the JMS listening connector:

1. Open a browser and enter the following URL:

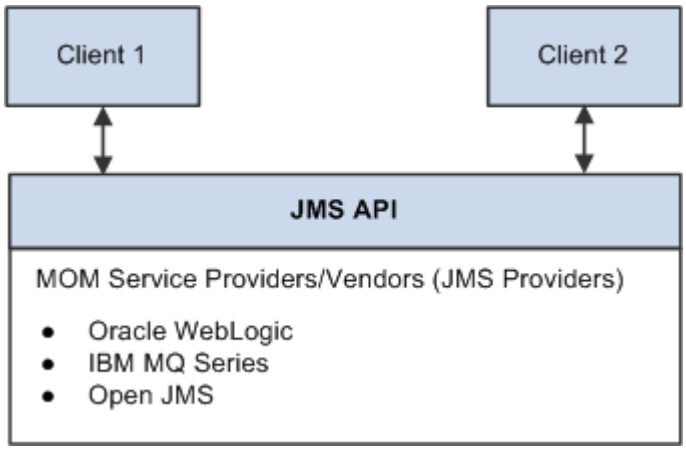
   http://localhost:port/PSIGW/JMSListeningConnectorAdministrator?Activity=Stop
2. Press Enter.

   Your web browser displays a message indicating that the JMS listening connector has stopped.

Note. You must register the JMSListeningConnectorAdministrator servlet object under the web application PSIGW in the web server. Your web server documentation should provide instructions about how to register a servlet. The class name is com.peoplesoft.pt.integrationgateway.listeningconnector.JMSListeningConnectorAdministrator.

Using the JMS Target Connector

JMS is an application programming interface (API) for accessing message systems. JMS provides a standard Java-based interface to the message services of message-oriented middleware (MOM) providers. The JMS target connector is an adapter to JMS providers, and it can be used with MOM and JMS providers, such as Oracle Weblogic, IBM MQSeries and others. The following diagram illustrates how messages flow through the JMS API:



Message flow through the JMS API

The primary features of JMS are.

- Connection factories that are used to create connections to a specific JMS provider.
- Separate publish, subscribe, and point-to-point messaging domains.

  These are defined by separate interfaces so that a provider does not have to support both.
- Topics for publish and subscribe, as well as queues for point-to-point messaging.

When multiple applications must receive the same message, publish and subscribe messaging is used. In publish and subscribe messaging, all of the subscribers subscribe to a topic and all of the publishers publish messages to a topic. The messaging system distributes messages from the publisher to the subscriber. This domain is mainly used for asynchronous messaging.

When one application must send a message to another application, point-to-point messaging is used. This domain is only for synchronous messaging. There are two basic types of point-to-point messaging systems. One uses a client that directly sends a message to another client. The other, more commonly used implementation uses a message queue.

The JMS target connector either publishes a message to a topic or inserts a message into a queue, based on the node-level properties that you set.

The JMS target connector supports only JNDI file context for the lookup of connection factories, topics, and queue names. (Lightweight Directory Access Protocol (LDAP) is not supported.)

The connector ID for the JMS target connector is *JMSTARGET*.

Asynchronous and Synchronous Communication

The JMS target connector provides both synchronous and asynchronous modes of communication. When the node level property ReplyTo is set to *False,* communication is asynchronous. When it is set to *True,* communication is synchronous.

For asynchronous communication, the JMS target connector publishes messages to MOM or drops messages into a queue and commits the session. It does not wait for a response from the destination system. For synchronous communication, after the connector publishes messages or drops them into a queue, it waits for the temporary topic or queue to respond.

For synchronous communication, the exchanges involve only the publisher and a single subscriber. When a JMS-compliant remote node receives a synchronous request message from PeopleSoft, it must use the value of the message ID of the request message to populate the correlation ID of its response message. When the response is received by the PeopleSoft JMS target connector, it compares the JMS correlation ID of the response message with the JMS message ID of the request. The message is not accepted if these two IDs do not match.

When sending messages either synchronously or asynchronously, the connector sets different string properties in the JMS message header. The properties are used as metadata about the message. The JMS target connector also sets a reference to the temporary queue or topic from which it requires the response.

JMS Target Connector and Message Segments

The JMS target connect is segment-aware and you may use it to send message segments to integration partners.

See *Enterprise PeopleTools 8.50 PeopleBook: PeopleSoft Integration Broker*, "Sending and Receiving Messages," Working With Message Segments.

IBInfo Data Contained in JMS Headers

A message has two parts—the transaction data and the IBInfo header that is the routing envelope used by PeopleSoft Integration Broker. In the event that a receiving system wants to make use of the IBInfo data, IBInfo header information is included when publishing messages to non-PeopleSoft systems when using the JMS target connector or the HTTP target connector.

When using the JMS target connector to send messages to non-PeopleSoft systems, the following IBInfo data is contained in the JMS headers. The content of the message (message body) is not impacted.

- RequestingNode
- FinalDestinationNode
- DestinationNodes
- MessageName
- MessageType
- OrigTimeStamp
- NonRepudiation

Gateway-Level Connector Properties

There are no gateway-level JMS target connector properties that you must set.

Node-Level Connector Properties

You must set either a JMS queue or JMS topic for a given node definition. If both are set or are missing the PeopleSoft Integration Broker generates an invalid message exception.

Note. You must register JMS-administered objects—such as topics, queues, and connection factories—that you include as connector properties. The documentation for specific providers should provide instructions on how to register the topics.

JMS message types can be *Text,Map Message,Stream,* or *Object.* However, PeopleSoft provides only text messages. If you need to use other message types, you can write a class that implements the com.peoplesoft.pt.integrationgateway.common.jms.ProcessJMSMessage interface, and you set the class name as a value for JMSMessageTypeClass.

The provider name that you specify for the JMSProvider in the node definition must match the JMSProvider.JNDIFactory property that you specify in the integrationGateway.properties file.

The following table describes the node-level connector properties:

Property ID	***Property Name***	***Description***
JMSTARGET	JMSAcknowledgement	Specify the acknowledgment type. Values are: • *Auto_Acknowledge.* This is the default value. • *Client_Acknowledge.*
JMSTARGET	JMSDeliveryMode	Specify either durable or nondurable delivery. Values are: • *Persistent.* • *Non-persistent* (default).
JMSTARGET	JMSFactory	Specify the factory name. The default value is *QueueConnectionFactory.*
JMSTARGET	JMSMessageTimeToLive	Specify the time in seconds.
JMSTARGET	JMSMessageType	Specify the type of message to send. Values are: • *Text* (default). • *MapMessage.* • *Stream.* • *Object.*

Property ID	Property Name	Description
JMSTARGET	JMSMessageTypeClass	(Optional.) Specify the implementation class of ProcessJMSMessage. You must set this property when the JMSMessageType is anything other than *Text*.
JMSTARGET	JMSPassword	(Optional.) Specify the password to access the connection. If you choose to specify a password, you must encrypt it. See Chapter 5, "Managing Integration Gateways," Encrypting Passwords, page 44.
JMSTARGET	JMSPriority	Specify the message priority for delivery. Values range from *0* to *9*. A value of *9* indicates the highest priority. The default is *0*.
JMSTARGET	JMSProvider	Specify the JMS provider's name. Values are: • *MQSeries* (default). • *WebLogic*.
JMSTARGET	JMSQueue	(Optional.) Specify the queue name, if you use a queue. You must use and specify either a topic or a queue.
JMSTARGET	JMSReplyTo	Set this property to *True* to receive a response from the external system. Values are: • *True*. • *False* (default).
JMSTARGET	JMSTopic	(Optional.) Specify the topic name, if you use a topic. You must use either a topic or a queue.
JMSTARGET	JMSUrl	Specify the URL.

Property ID	Property Name	Description
JMSTARGET	JMSUserName	(Optional.) Specify the username to establish a connection to the JMS.
JMSTARGET	JMSWaitForResponse	Specify the time in milliseconds for the connector to wait for the temporary response queue to return a synchronous response message. If a response fails to appear in the queue within the specified period, the transaction fails and the queue is deleted. The default value is *60000* (60 seconds).
JMSTARGET	SOAPUpContent	(Optional.) Automatically wrap outbound transactions in SOAP format. The valid values are: • Y. Outbound messages are wrapped in SOAP format. • N. Outbound messages are not wrapped in SOAP format.
HEADER	SendUncompressed	Specify whether to send messages decompressed. Values are: • *Y:* Send the message decompressed and unencoded. This is the default value. • *N:* Send the message compressed and base 64 encoded.
HEADER	SOAPAction	(Optional.) Enable third-party systems (for example, Universal Description, Discovery, and Integration (UDDI) sites) to receive SOAP transactions over HTTP. The default value is *""* (a null string).

Additional Setup Steps

Before using the JMS target connector, verify that:

1. The JMS messaging system is running.
2. All JMS connection factories, topics, and queues are registered for JNDI lookup.
3. A username and a password are created in the JMS system for use as values for the properties JMSUserName and JMSPassword.

JMS Target Connector Errors and Exceptions

The JMS target connector may generate the following exceptions:

Exception	*Cause*
InvalidMessageException	This exception is generated when any node level or connector parameters are not set properly. Examples are: • Both queue and topic are specified. • Neither queue nor topic is specified. • A JMS security exception is generated. (Verify that the username and password are correct.) • A naming exception occurs.
ExternalApplicationException	This exception is generated when: • The correlation ID does not match when the ReplyTo property is set to *True.* • The message could not put into a queue, or a topic could not be published.
GeneralFrameWorkException	This exception is generated when a naming exception occurs.

Adding Generic JMS Providers

The JMS providers that PeopleSoft supports are Oracle WebLogic, and IBM MQSeries. However, to meet your business requirements you can add generic JMS providers.

This section provides lists of configuration tasks to perform on the JMS listening connector and JMS target connector to add a generic JMS provider to PeopleSoft Integration Broker.

Configuring the JMS Listening Connector for Generic JMS Providers

To configure the JMS listening connector for a generic JMS provider:

- Obtain the following information from the provider:
  - JMS jar file.
  - JNDIFactory information
- Determine if messaging will be in topics or queues.
- Determine if error handling will be in topics or queues.

- Update JMS properties in the integrationGateway.properties file:
  - Update the JNDIFactory entry.

    For example if the provider were Tibco the entry might be:

    ```
    ig.jms.JMSProvider.JNDIFactory.Tibco=com.tibco.JMSFactory
    ```
  - Populate the appropriate messaging topic and queue entries based on how messaging will be handled.
  - Populate the appropriate error topic and queue entries based on how messaging will be handled.

In addition to the information provided in this section, review the JMS Headers Properties section of this chapter which discusses the required information that must be in the headers of each message processed by the JMS listening connector.

Configuring the JMS Target Connector for Generic JMS Providers

To configure the JMS target connector for a generic JMS provider:

- Define a node for the provider.
- Assign the JMS target connector to the provider node and specify the target connector properties.

Working With the Simple File Target Connector

This section discusses the simple file target connector.

Understanding the Simple File Target Connector

With the simple file target connector, you can save PeopleSoft messages as files in XML format. This enables you to verify that:

- You have composed messages correctly.
- The messages contain the content that you want to send.

You can use the Simple File target connector to send messages using the PUT command and receive messages using the GET command.

The connector ID for the simple file target connector is *FILEOUTPUT*.

Setting File Security

To secure files during processing, set the property ig.fileconnector.password in the integrationGateway.properties file, in addition to the Password property in the connector properties set in the Gateways component.

Setting file security is optional. However, if you use this feature, both passwords must match and be encrypted.

See Chapter 5, "Managing Integration Gateways," Encrypting Passwords, page 44.

Node-Level Connector Properties

The following table describes the node-level connector properties:

Property ID	*Property Name*	*Description*
HEADER	SendUncompressed	Specify whether to save messages decompressed. Values are: • *Y:* Save the message decompressed and unencoded. This is the default value. • *N:* Save the message compressed and base64 encoded.
PROPERTY	FilePath	Specify the location where the connector saves the output file. The default location is c:\temp.
PROPERTY	FileName	(Optional.) Specify the name of the output file. The file's default name has the following format: `<sourcenodename>.<operationname>.<segmentid>.xml` If the outbound message has multiple segments, each segment is saved as an individual file and each file is appended with its segment ID.
PROPERTY	Method	Specify the method used to send messages. The valid values are: • *PUT.* (Default.) • *GET.*
PROPERTY	Password	(Optional.) Specify a password for secure processing. For secure processing, you must also set the ig.fileconnector.password in the integrationGateway.properties file. See the Setting File Security section earlier in this chapter.

Working With the FTP Target Connector

This section discusses working with the FTP target connector.

Understanding the FTP Target Connector

The FTP target connector enables the gateway to use FTP to send messages to and receive messages from FTP servers. It uses the PUT command to place messages or files from the integration gateway onto remote FTP servers. The GET command is used to receive messages from FTP servers. Outbound messages through the FTP target connector are UTF-8 encoded.

Note. The FTP target connector handles string-based data only. Binary data is not natively supported in PeopleSoft Integration Broker.

PeopleSoft Integration Broker also supports secure communication with FTP servers using FTPS.

The connector ID for the FTP target connector is *FTPTARGET*.

Prerequisites for Using the FTP Target Connector

In addition to specifying Java Archive (JAR) files in the web server CLASSPATH and setting node-level connector properties, to use this connector you must also specify the integration gateway URL in the Gateways component. Information about specifying the required JAR files and setting node-level FTP and FTPS connector properties is discussed in this section.

Specifying Required JAR Files

For the FTP target connector to function properly the following JAR files from IBM must reside in the CLASSPATH of the web server running the integration gateway:

- FTPProtocol.jar
- ipworksssl.jar (required for FTPS)

Setting Node-Level FTP Connector Properties

This section describes the required node-level properties you must set to use the FTP target connector.

The following table describes the required node-level connector properties:

Property ID	*Property Name*	*Description*
HEADER	SendUncompressed	Specify whether to send messages decompressed. Values are: • *Y:* Send messages decompressed and decoded. This is the default value. • *N:* Send messages compressed and base64 encoded.
FTPTARGET	HOSTNAME	Specify the IP address or name of the FTP server for the connection.
FTPTARGET	METHOD	Specify the method to send or receive messages. The valid values are: • PUT (default). Send messages to an FTP server. • GET. Retrieve messages from an FTP server. • GETDIRLIST. Retrieve a directory list of files from an FTP server.
FTPTARGET	DIRECTORY	Specify the remote directory into which the file is placed. **Note.** When using the GET method you must specify the location where the file resides for the method to function properly. If not specified, the default directory of the FTP server on the remote site is used.

Property ID	*Property Name*	*Description*
FTPTARGET	FILENAME	(Optional.) Specify the name of the file saved on the recipient's FTP server. By default, the file name is a concatenation of the following: • Originating node name. • Originating username. • Operation name. • Originating timestamp. • Segment ID. If you do not specify a filename, the FTP(S) target connector performs a GET to retrieve the directory list from the remote FTP server. See the section on Directory List Support earlier in this section.
FTPTARGET	USERNAME	Enter the FTP server login ID.
FTPTARGET	PASSWORD	Enter the password for the login to the FTP server. This password must be encrypted. See Chapter 5, "Managing Integration Gateways," Encrypting Passwords, page 44.
FTPTARGET	TIMEOUT	Specify the time in milliseconds for the connector to wait for the message to transmit. If the timeout period expires without a successful transmission, the transaction fails. The default value is *50000* (50 seconds).
FTPTARGET	TYPE	Indicates the FTP mode used to transfer the file. The valid options are: • ASCII (default) • BINARY When you select *ASCII*, all characters are converted to their ASCII equivalents. When you select *BINARY*, data is copied bit-by-bit and no conversion is performed.

Setting Node-Level FTPS Connector Properties

The following table describes properties to use for secure FTPS communication.

Property ID	*Property Name*	*Description*
FTPTARGET	FTPS	Enables secure communication over FTP. Values are: • *Y:* Enable FTPS communication. • *N:* Disable FTPS communication. This is the default value.
FTPTARGET	CLIENTCERT	(Optional.) To use client authentication when establishing a connection with the target or receiving system, enable the CLIENTCERT property.
FTPTARGET	PORT	Specify the port used for communication. The default port is *21*.
FTPTARGET	SSLSTARTMODE	(Optional). Use this property to set the SSL start mode. Values are: • *DEFAULT.* If the remote port is set to the standard plain text port of the protocol (where applicable), it will behave the same as if SSLSTARTMODE is set to *sslExplicit* . In all other cases, SSL negotiation will be implicit *(sslImplicit).* • *IMPLICIT.* The SSL negotiation will start immediately after the connection is established. • *EXPLICIT.* The connector first connects in plain text, and then explicitly starts SSL negotiation through a protocol command such as STARTTLS.

Using Directory Lists

One of the optional node-level FTP connector properties is FILENAME. If you do not know the file name of the file you would like to receive but do not know the directory in which it resides, you can use the GETDIRLIST method to retrieve a directory list. The directory list is retrieved in XML format and you must parse the XMLDocument to read its contents. You can then use the GET method to get the actual file. The following example shows the format of a returned directory list.

```
<DirList>
   <File name="sample.bat">
      <Date></Date>
      <Size>1234</Size>
      <Time></Time>
      <isFile>True</isFile>
   </File>
   <File name="sample2.bat">
      <Date></Date>
      <Size>1234</Size>
      <Time></Time>
      <isFile>True</isFile>
   </File>
   <File name="temp">
      <Date></Date>
      <Size>1234</Size>
      <Time></Time>
      <isFile>False</isFile>
   </File>
</DirList>


Date : Date on the file on remote system
Time : Time on the file on remote system
Size : Size of the file
isFile : True if it is a file. False if it is a directory.
```

Directory List Example

The following example shows the code needed to use the FTP connector to get a list of the files in a directory, run through the list of files, select a file, and retrieve it. To use this example, you must know the directory in which the file resides.

If you know the name of the file you wish to receive but do not know the directory, use the FILENAME property and the GETDIRLIST method to retrieve a directory list, as described previously in this section.

See Chapter 6, "Using Listening Connectors and Target Connectors," Using Directory Lists, page 111.


```
Local XmlDoc &Output;
Local Message &MSG1, &MSG2, &MSG3;

&MSG = CreateMessage(OPERATION.QE_FLIGHTPLAN_UNSTRUCT);

/* Set ConnectorName and Connector ClassName */
&MSG.IBInfo.IBConnectorInfo.ConnectorName = "FTPTARGET";
&MSG.IBInfo.IBConnectorInfo.ConnectorClassName = "FTPTargetConnector";

/* Set the FTP connector properties in the ConnectorInfo */
/* Method name can be either Get or GetDirlist.  */
&nRet = &MSG.IBinfo.IBConnectorInfo.AddConnectorProperties("Method",
     "GET",%Property);
&nRet = &MSG.IBinfo.IBConnectorInfo.AddConnectorProperties("HOSTNAME",
     "ftp.ftpserver.com",%Property);
&nRet = &MSG.IBinfo.IBConnectorInfo.AddConnectorProperties("USERNAME",
      "sam",%Property);

/*  Encrypt the password */
&pscipher = CreateJavaObject("com.peoplesoft.pt.integrationgateway.common.
     EncryptPassword");
&encPassword= &pscipher.encryptPassword("ftpserverpassword");
 &pscipher = Null;

&string_return_value = &MSG.IBinfo.IBConnectorInfo.AddConnectorProperties
     ("PASSWORD", encPassword,%Property,);
&string_return_value = &MSG.IBinfo.IBConnectorInfo.AddConnectorProperties
     ("DIRECTORY", "/incoming/tmp",);

/* Do Connector Request */
&MSG2 = %IntBroker.ConnectorRequest(&MSG);

/* Get XMLDoc from MSG2*/
&fileListXmlDoc = &MSG2.GetXmlDoc();

/*Parse the XMLDoc. Structure of the DirList Message is
<DirList>
     <File name="sample.bat">
        <Date></Date>
        <Size>1234</Size>
        <Time></Time>
        <isFile>True/False</isFile>
     </File>
</DirList>*/

&XmlNode = &fileListXmlDoc .DocumentElement.FindNode("File ");

/* Get the file name */
&fileName = &XmlNode.NodeValue

/* Get the file name from the Remote FTPServer */
&MSG = CreateMessage(OPERATION.QE_FLIGHTPLAN_UNSTRUCT);

/* Set ConnectorName and Connector ClassName */
&MSG.IBInfo.IBConnectorInfo.ConnectorName = "FTPTARGET";
&MSG.IBInfo.IBConnectorInfo.ConnectorClassName = "FTPTargetConnector";

/* Set the FTP connector properties in the ConnectorInfo */
/* Mehtod name can be either Get */
&nRet = &MSG.IBinfo.IBConnectorInfo.AddConnectorProperties("Method",
   "GET",%Property);
&nRet = &MSG.IBinfo.IBConnectorInfo.AddConnectorProperties("FILENAME",
     &fileName,%Property);
&nRet = &MSG.IBinfo.IBConnectorInfo.AddConnectorProperties("HOSTNAME",
```



```
     "ftp.ftpserver.com",%Property);
&nRet = &MSG.IBinfo.IBConnectorInfo.AddConnectorProperties("USERNAME",
   "sam",%Property);

/*  Encrypt the password */
&pscipher = CreateJavaObject("com.peoplesoft.pt.integrationgateway.common.
     EncryptPassword");
&encPassword= &pscipher.encryptPassword("ftpserverpassword");
 &pscipher = Null;

&nRet = &MSG.IBinfo.IBConnectorInfo.AddConnectorProperties("PASSWORD",
     encPassword,%Property,);
&nRet = &MSG.IBinfo.IBConnectorInfo.AddConnectorProperties("DIRECTORY",
   "/incoming/tmp",);

/* Do Connector Request */
&MSG3 = %IntBroker.ConnectorRequest(&MSG);
```

Working With the AS2 Connectors

This section provides an overview of the Applicability Statement 2 (AS2) specification and discusses how to:

- Work with the AS2 listening connector.
- Work with AS2 response connector.
- Work with the AS2 target connector.

Understanding Using AS2

AS2 is specification for Electronic Data Interchange (EDI) between organizations using the internet. AS2 uses Secure/Multipurpose Internet Mail Extensions (S/MIME), which secures data with authentication, nonrepudiation and encryption. The transportation protocol for this specification is HTTP and HTTPS for real-time communication. S/MIME secures data with authentication, message integrity and nonrepudiation.

PeopleSoft Integration Broker provides three connectors for use with AS2:

AS2 listening connector Use the AS2 listening connector to receive request messages in AS2 format.

AS2 response connector The AS2 response connector sends acknowledgements for data you receive from the AS2 listening connector.

AS2 target connector Use the AS2 target connector to send messages in AS2 format.

You can use the AS2 listening and target connectors to transport any kind of data, including, but not limited to, XML, EDI, text and binary data.

The AS2 target connect is segment-aware and you may use it to send message segments to integration partners.

See *Enterprise PeopleTools 8.50 PeopleBook: PeopleSoft Integration Broker*, "Sending and Receiving Messages," Working With Message Segments.

Understanding MDNs

AS2 uses two different message types: the request message containing the data to be integrated and the Message Disposition Notification (MDN) to acknowledge the receipt of the data.

AS2 message exchange can occur over HTTP or HTTPS. The sender must request and MDN from the receiver, that enables the sender to verify that the message has been transferred in an unmodified state and that the receiver has been able to decompress or decrypt the message.

As an option, an MDN may be digitally signed, enabling the recipient to authenticate the sender of the MDN to check the integrity of the incoming message.

MDNs can be delivered synchronously or asynchronously.

Synchronous MDNs

Synchronous MDNs are returned to the sender in the same HTTP connection that sent the message. Processing does not continue until the sender receives the MDN.

Asynchronous MDNs

Asynchronous MDNs are delivered to the sender at a later time after the transmission of the message.

AS2 Requests initiated by the AS2 target connector with an asynchronous MDN Type must send MDN asynchronous responses to the AS2 response connector at the following URL:

```
http://<SERVER><PORT>/PSIGW/AS2ResponseConnector
```

The AS2 response connector processes MDNs by verifying them with sent request and publishes a response message to the PeopleSoft Integration Broker.

When a message is published the AS2 target connector stores the information regarding the request (for example, MessageID, signed algorithm, and so forth) for verifying the response on the integration gateway. When the response is received, the AS2 response connector verifies with the request information and publishes a response message to PeopleSoft Integration Broker.

A published asynchronous response is an empty message with the following structure:

```
<? Xml version="1.0">
<AS2ASyncResponse>
      <ConversationID>123213</ConversationID>
      <OriginalMessageID>23234<OriginalMessageID>
      <MDN>123123 . . </MDN>
      <ReceiptMessage> <![CDATA[Receipt message.]]></ReceiptMessage>
      <MDNVerified>True/False<MDNVerified>
</AS2ASyncResponse>
```

PeopleSoft Integration Broker generates the conversation ID tag when a message is published. This tag is used to correlate the MDN with the request message.

If the MDNVerified tag is set to *True*, the integration gateway has successfully verified the MDN.

Note. To provide application the flexibility to take appropriate action with responses and response status information, it is the developer's responsibility to write subscription PeopleCode for processing acknowledgement messages and correlating them with requests. Without subscription PeopleCode to consume the message, an MDN will not be sent back to the source.

The AS2 connectors implement correlation IDs in MDNs. The AS2 target connector saves the outbound message ID as a correlation ID in the directory defined in the ig.AS2.AS2Directory in the integrationGateway.properties file .

When the response arrives later, the AS2ResponseConnector checks the conversationID from the response message with the one saved by early. If they don't match, the transaction fails.

PeopleCode Considerations

In outbound messages, always use the %Intbroker.publish () function. Using %IntBroker.SyncRequest results in errors.

Understanding the AS2 Listening Connector

The AS2 listening connector can receive inbound asynchronous request messages, and can send synchronous and asynchronous MDNs. This section describes how these messages flow through the AS2 listening connector and how MDNs are created and returned to the senders of messages.

Inbound Asynchronous Request—Synchronous MDN

This section describes the process flow of an inbound asynchronous request message through the AS2 listening connector, with the integration engine generating a synchronous MDN.

1. The AS2 listening connector receives an AS2 message over an open HTTP connection.
2. The connector verifies the digital signature and decrypts the message. If necessary, the connector also decompresses the message.
3. The AS2 listening connector sends the message to the integration engine.
4. The integration engine creates an MDN and sends it back to the integration gateway as part of the HTTP response message.

Inbound asynchronous Request—Asynchronous MDN

This section describes the process flow of an inbound asynchronous request message through the AS2 listening connector, with the integration engine generating an asynchronous MDN.

1. The AS2 listening connector receives a message over HTTP.
2. The AS2 listening connector closes the connection and sends a status code of 200.
3. The connector verifies the digital signature and decrypts the message. If necessary, the connector also decompresses the message.
4. The AS2 listening connector sends the message to the integration engine.

5. The integration engine creates an MDN and sends it back to the sender as an asynchronous transaction, using the AS2 target connector.

Understanding the AS2 Response Connector

When a request is published, PeopleSoft Integration Broker generates a conversation ID in the message ID field of the request message. Then, when an MDN comes back it extracts the conversation ID from the message to correlate the MDN acknowledgement with the request message.

Note. You must write subscription PeopleCode to process acknowledgement messages and to correlate them with requests messages. This provides flexibility for you to specify actions to take based on response status.

When it receives an MDN, the AS2 response connector checks for the conversation ID, constructs the asynchronous response message by setting the conversation ID, MDN, and the message/subject received with the MDN. It then sends the response to the integration engine.

Understanding the AS2 Target Connector

This section describes how messages flow through the AS2 target connector and how the connector processes MDNs.

Note. The AS2 target connector sends message requests in asynchronous mode only. However, the connector can receive MDNs in synchronous or asynchronous mode.

Outbound Asynchronous Request—Synchronous MDN

This section describes the process flow of outbound asynchronous request message through the AS2 listening connector, with the integration engine generating a synchronous MDN.

1. The AS2 target connector receives the request message from the integration engine.
2. The AS2 target connector checks the outbound message to determine if an MDN is required, and if so, whether the MDN is synchronous or asynchronous.
3. The AS2 connector makes an HTTP request to the receiver.
4. The AS2 connector verifies the MDN in the HTTP response if an MDN is requested.
5. Once the MDN is verified, the AS2 connector sends a response to the integration engine indicating whether the message was sent successfully.

Outbound Asynchronous Request—Asynchronous MDN

This section describes the process flow of an outbound synchronous request message through the AS2 listening connector, with the integration engine generating an asynchronous MDN.

1. The AS2 target connector receives the request message from the integration engine.
2. The AS2 target connector checks the outbound message to determine if an MDN is required, and if so, whether the MDN is synchronous or asynchronous.

3. Check for MDNAsynchronousURL and request a Asynchronous Receipt (MDN).
4. The AS2 connector makes an HTTP request to the receiver.
5. The AS2 connector reads the HTTP status code and sends a response to the integration engine indicating whether the message was sent successfully.
6. At a later time, the AS2 listening connector receives an MDN from the receiver. The MDN is then processed.

   See Chapter 6, "Using Listening Connectors and Target Connectors," Understanding MDNs, page 114.

Using the AS2 Listening Connector

This section describes how to use the AS2 listening connector and discusses how to:

- Set required header parameters.
- Set optional header parameters.
- Set gateway-level properties.

Setting Required Header Parameters

The following HTTP header parameters are require in incoming AS2 requests:

HTTP Header Parameter	*Description*
AS2From	Specify the name of the sending node.
AS2To	Specify the name of the receiving node.

If the AS2From and AS2To node names are not PeopleSoft node names, you must map them in the integrationGateway.properties file.

Setting Optional Header Parameters

When using the AS2 listening connector, you may set the following optional HTTP header parameters:

HTTP Header Parameter	*Description*
MessageName	(Optional.) Specify the name of the incoming operation or message. **Note.** You can specify the message name in the HTTP header, HTTP query string or in the integrationGateway.properties file.
Password	(Optional.) Specify an encrypted password for node authentication.
MessageVersion	(Optional.) Specify the version of the message. If you specify a message name in the MessageName parameter, enter the message version.

HTTP Header Parameter	Description
OrigUser	(Optional.) Specify the username of the originating user.
ExternalMessageID	(Optional.) Specify a unique ID that identifies the message. If two messages are published with the same external message ID, the first message is processed and the second messages is marked as a duplicate.

Setting Gateway-Level Properties

To configure the AS2 listening connector, you must set properties located in the integrationGateway.properties file for each message the connector receives.

Note. Replace text in angle brackets (for example <project_branch>) with the appropriate values.

Property	Description
ig.AS2.LogDirectory	(Optional.) Specify the directory to log all incoming and outgoing AS2 requests and responses. For example: ig.AS2.LogDirectory = c://temp//as2//logs
ig.AS2.KeyStorePath	Specify the path to the Java keystore. For example: C://pt850 //webserv//peoplesoft//keystore//pskey
ig.AS2.KeyStorePassword	Specify the encrypted password to the Java keystore. For example: GD9klUFw8760HVaqeT4pkg==
ig.AS2.AS2ListenerMap.From.<from alias>	(Optional.) If a sending or receiving node is not a PeopleSoft node, you must map it in the integrationGateway.properties file. Use this property if the sending system is not a PeopleSoft node. Replace the information in brackets with an alias of the sending system and set it equal to the remote node name in the PeopleSoft application database. For example: ig.AS2.AS2ListenerMap.From.QE_SOURCE= PT_LOCAL
ig.AS2.AS2ListenerMap.To.<to alias>	(Optional.) If a sending or receiving node is not a PeopleSoft node, you must map it in the integrationGateway.properties file. Use this property if the receiving system is not a PeopleSoft node. Replace the information in brackets with an alias of the receiving system and set it equal to the remote node name in the PeopleSoft application database. For example: ig.AS2.AS2ListenerMap.To. QE_IBTGT= AS2TARGETNODE

Property	Description
ig.AS2.<source>.<target>.CertificateAlias	Specify the certificate (target) alias name. Replace <source> and <target> with the source and target PeopleSoft node names used in the AS2FROM and AS2TO HTTP headers, or those mapped in the properties above. For example: ig.AS2.PT_LOCAL.AS2TARGETNODE.CertificateAlias=JFRANCO030204
ig.AS2.<source>.<target>.SignerCertificateAlias	Specify the certificate alias (source) used for signing the certificate. For example: ig.AS2.PT_LOCAL.AS2TARGETNODE.SignerCertificateAlias=JRICHAR2030104
ig.AS2.<source>.<target>.MessageName	(Optional.) Specify the name of the incoming message. Replace <source> and <target> with the source and target PeopleSoft node names used in the AS2FROM and AS2TO HTTP headers, or those mapped in the properties above. For example: ig.AS2.PT_LOCAL.AS2TARGETNODE.MessageName=EXAMPLE_REQUEST_MSG **Note.** You can specify the message name in the HTTP header, HTTP query string or in the integrationGateway.properties file.

Using the AS2 Target Connector

This section describes using the AS2 target connector and discusses how to:

- Set node-level connector properties.
- Set gateway-level connector properties.

Setting Node-Level Connector Properties

The following table lists the required and optional AS2 target connector properties you set at the node level. You set these properties in the Gateways component in the PeopleSoft Pure Internet Architecture.

Property ID	Property	Description
AS2PROPERTY	AS2From	Specify the name of the sending node.
AS2PROPERTY	AS2To	Specify the name of the receiving node.

Property ID	Property	Description
AS2PROPERTY	AsynchronousMDNRecipientURL	Specify a URL that indicates how and where the MDN is delivered. For example: http://<source webserver>:<http port>/PSIGW/AS2ResponseConnector By specifying a valid URL you can request asynchronous delivery instead. The URL indicates the destination for the reply, and may use any appropriate protocol, such as HTTP or HTTPS. If this property is set to an empty string (Default), the receipt is returned synchronously within an HTTP reply.
AS2PROPERTY	Compression	Specify whether to compress outbound AS2 messages. Options are: • *Y*: Send messages compressed using the Zlib compression format. • *N*: No compression. (Default.)
AS2PROPERTY	EDIType	Specify the content type of the message. Options are: • *Application/edi-x12.* • *Application/edifact.* • *Application/xml..* • *Application/text.*
AS2PROPERTY	EnableCRLF	(Optional.) PeopleSoft Integration Broker automatically removes carriage returns in messages and retains line feeds. Use this property to specify whether to add a carriage return (CR) back to the end of a line feed (LF). Options are: • *Y.* Adds CR to LF. (Default.) • *N.* No CR added to LF.
AS2PROPERTY	EncryptingAlgorithm	(Optional.) Specify the algorithm used to encrypt data. The default value is *3DES.* Use of this algorithm is highly recommended. When you specify an encrypting algorithm, you must set the RecipientCertAlias to a valid certificate. The data is encrypted using the RecipientCertAlias value you define with the algorithm you specify here.
AS2PROPERTY	FirewallHost	(Optional.) If connecting through a firewall, specify the firewall host name or IP address.

Property ID	Property	Description
AS2PROPERTY	FirewallPassword	(Optional.) If connecting through a firewall, specify an encrypted password if authentication is to be used when connecting through the firewall.
AS2PROPERTY	FirewallPort	(Optional.) If connecting through a firewall, specify the port of the firewall to which to connect. See the description for the FirewallType property for guidelines on how the default setting is made.
AS2PROPERTY	FirewallType	(Optional.) If connecting through a firewall, specify the type of firewall. Options are: • *NoFirewall.* (Default.) • *TunnelingProxy*: Connects through a tunneling proxy. The FirewallPort property is automatically set to *80.* • *SOCK4Proxy*: Connects through a SOCKS4 proxy. The FirewallPort property is automatically set to *1080.* • *SOCK5Proxy*Connects through a SOCKS5 proxy. The FirewallPort property is automatically set to *1080* You can overwrite port numbers in the FirewallProperty field.
AS2PROPERTY	Firewall User	(Optional.) If connecting through a firewall, specify the firewall user name if authentication is to be used connecting through a firewall.
AS2PROPERTY	Http Password	(Optional.) Specify the HTTP username if HTTP authentication is to be used.
AS2PROPERTY	HttpUser	(Optional.) Specify the HTTP username password if HTTP authentication is to be used.
AS2PROPERTY	MDNSecurityType	(Optional.) Specify the algorithm to use for signing the MDN. Options are: • *Signed-sha1*. (Default.) • *Signed-md5.* • *Unsigned.*
AS2PROPERTY	MDNType	Specify whether to generate an MDN, and if so the type to generate. Options are: • *None.* • *Sync*: Synchronous. (Default.) • *Async*: Asynchronous.
AS2PROPERTY	ProxyPassword	(Optional.) Specify the proxy user password.

Property ID	Property	Description
AS2PROPERTY	ProxyPort	(Optional.) Port of the proxy server to which to connect.
AS2PROPERTY	ProxySSL	(Optional.) Options are: • *Automatic.* (Default.) • *Always.* • *Never.* • *Tunnel.*
AS2PROPERTY	ProxyServer	(Optional.) Specify the proxy server name or IP address.
AS2PROPERTY	ProxyUser	(Optional.) Specify the user name if authentication is to be used to connect through a proxy.
AS2PROPERTY	RecipientCertAlias	(Optional.) Specify the alias name of the recipient's certificate. **Note.** This property is required if the EncryptingAlgorithm property is set.
AS2PROPERTY	SecurityType	Specify the security type of the request message. Options are: • *EncryptedOnly.* • *Signed-Encrypted.* (Default.) • *SignedOnly.* • *None.*
AS2PROPERTY	SignersCertificateSubject	Specify the alias name of the signing certificate. This property is required if the SecurityType property is set to *SignedOnly* or *Signed-Encrypted.*
AS2PROPERTY	TimeOut	(Optional.) Specify the timeout for the connector in seconds. When this value is set to *0*, all operation will run uninterrupted until successful completion, or an error condition is encountered. The default value is *60.*
AS2PROPERTY	User Agent	(Optional.) Specify the name of the user agent or email address.
BACKUPURL	URL	(Optional.) Specify the backup URL to use to send messages if delivery to the primary URL fails.
PRIMARYURL	URL	Specify the URL to which messages are sent using this connector.

Property ID	Property	Description
HEADER	sendUncompressed	Specify whether to send messages decompressed. Options are: *Y*: Send messages decompressed and decoded. (Default.) *N*: Send messages compressed and base64 encoded. **Note.** Do not change the default value.
PRIMARYURL	URL	Specify the URL to which messages are sent using this connector. For example: http://<target webserver>:<http port>/PSIGW/AS2ListeningConnector

Setting Gateway-Level Connector Properties

This section describes required AS2 target connector properties you set in the integrationGateway.properties file.

The AS2 target connector uses digital certificates for digital signatures, nonrepudiation and encryption.

As a result, you must set up digital certificates to use the connector.

Public keys and signatures are stored in certificates, so there must be a place in the organization to store these keys and certificates.

The place to store keys is the key store. A key store can be a flat file, a database or an LDAP server that can store key material. PeopleSoft keystore is installed with the PeopleSoft Pure Internet Architecture at the following default location: <PIA_HOME>\webserver\<DOMAIN>\keystore. PeopleSoft AS2 connectors will invoke these certificates from JKS. JKS exists on the web server.

The following properties should be set in the integrationGateway.properties file of the source web server in order to use the AS2 target connector. Use the PSCipher utility to encrypt the password.

Property	Description
ig.AS2.KeyStorePath	Specify the path to the Java keystore. For example: C://pt850//webserv//peoplesoft//keystore//pskey
ig.AS2.KeyStorePassword	Specify the encrypted password to the Java keystore. For example: GD9klUFw8760HVaqeT4pkg==
ig.AS2.AS2Directory	Specify the directory to log MDN responses. This property is required for asynchronous MDNs. For example: c://temp//as2

Property	Description
ig.AS2.LogDirectory	(Optional.) Specify the directory to log all incoming and outgoing AS2 requests and responses. For example: c://temp//as2//logs

Working With the SMTP Target Connector

This section provides an overview of the SMTP target connector and discusses how to:

- Set gateway-level connector properties.
- Set node-level connector properties.

Understanding the SMTP Target Connector

The SMTP target connector enables the gateway to send messages by email using SMTP. This connector supports plain text and HTML text content types. The connector supports the following fields: To:, From:, cc:, and bcc:. You can send data of any format in the body of the email.

You can include only one email address per type of address in the header. For instance, you can include only one addressee as a destination (DestEmailAddress).

The connector ID for the SMTP target connector is *SMTPTARGET*.

The SMTP target connect is segment-aware and you may use it to send message segments to integration partners.

See *Enterprise PeopleTools 8.50 PeopleBook: PeopleSoft Integration Broker*, "Sending and Receiving Messages," Working With Message Segments.

Setting Gateway-Level SMTP Target Connector Properties

The SMTP target connector has one gateway-level property, in the section of the integrationGateway.properties file labeled *DELIVERED CONNECTOR CONFIGURATION Section.* This property specifies the SMTP mail server host through which the connector sends messages. Specify the host as follows:

```
ig.connector.smtptargetconnector.host=SMTP_domain_name
```

Setting Node-Level SMTP Target Connector Properties

The following table describes the required node-level properties for the SMTP target connector:

Property ID	*Property Name*	*Description*
SMTPTARGET	SourceEmailAddress	Specify the email address from which you send messages. Only one address is currently allowed.
SMTPTARGET	DestEmailAddress	Specify the email address to which you send messages. Only one address is currently allowed.
SMTPTARGET	CC	(Optional.) Specify the email address of the party to which you copy messages. Only one address is currently allowed.
SMTPTARGET	BCC	(Optional.) Specify the email address of the party to which you send blind copies of messages. Only one address is currently allowed.
HEADER	Content-Type	(Optional.) Specify the type of text content that makes up the email body. Values are: • *Text/plain.* • *Text/html.*
HEADER	SendUncompressed	Specify whether to send messages decompressed. Values are: • *Y:* Send the message decompressed and unencoded. This is the default value. • *N:* Send the message compressed and base 64 encoded.

Chapter 7

Adding and Configuring Nodes

This chapter discusses how to:

- Add a node definition to the system.
- Configure a node.
- Rename or delete a node.

Understanding Nodes

Nodes represent any organization, application or system that will play a part in integrations.

For example, nodes can represent customers, business units, suppliers, other trading partners, external or third-party software systems, and so on.

Node definitions define the locations to or from which messages can be routed.

Because an application can send messages to itself, a default local node definition that represents the application is delivered as part of the integration engine.

Each PeopleSoft installation must have one, and only one, default local node

Local and Remote Nodes

Each PeopleSoft Integration Broker database involved in an integration must contain a default local node definition for itself, and a remote node definition for each of the other nodes involved.

Local and remote nodes are concepts relative to the database in which the nodes are defined. If you're signed on to Database A which has Node A defined, then Node A is local. If you're signed on to Database B, Node A is defined as remote.

For example, if the following definitions exist in the Node A database:

- NODE_A (default local)
- NODE_B (remote)

The following definitions must exist in the Node B database for it to integrate with Node A:

- NODE_A (remote)
- NODE_B (default local)

In practice, only portals use nodes designated simply as *Local*. The only local node definition used by PeopleSoft Integration Broker is the one designated *Default Local*, which represents the database onto which you are signed.

PeopleTools-Delivered Nodes

This section discusses nodes that are delivered with PeopleTools.

AIA Node

The *AIA* node is used for Oracle Application Integration Architecture (AIA) integrations, and represents an AIA integration partner.

Warning! Do not modify or delete the *AIA* node.

Anonymous Node

The *Anonymous* node is designated as the requesting node within PeopleSoft Integration Broker for third-party integrations that do not pass in a requesting node, but do have a defined any-to-local routing definition enabled on the service operation to be invoked.

Warning! Do not delete the *Anonymous* node.

You must modify the Anonymous node and define a Default User ID. The Default User ID that you specify is the ID that the system assigns to transactions that do not pass in a user ID.

Atom Node

The *Atom* node is used in association with PeopleTools feeds functionality.

You can use the *Atom* node only with asynchronous service operations. You cannot use the Atom node as the sending node. When the Atom node is the receiving node, the sending node must be the default local node.

Warning! Do not delete the *Atom* node.

Feeds are described elsewhere in PeopleBooks.

See *Enterprise PeopleTools 8.50 PeopleBook: Feed Publishing Framework*, "Oracle's PeopleSoft Feed Publishing Framework Preface."

BPEL Node

PeopleSoft delivers a node with the name *BPEL* specifically for integrations with BPEL process-based integrations when you are using Oracle BPEL Process Manager as the runtime engine. If you are using Oracle BPEL Process Manager, you must configure this node.

Information about configuring this node is described elsewhere in PeopleBooks.

See *Enterprise PeopleTools 8.50 PeopleBook: PeopleSoft Integration Broker*, "Integrating with BPEL Process-Based Services," Configuring the PeopleSoft-Delivered BPEL Node.

Default Local Node

The *Default Local Node* represents the system on which the application database is installed.

PeopleSoft Integration Broker is delivered with one node predefined as the default local node. You can't change which node is the default local node, but you can rename it to a more appropriate and meaningful name for your application or system.

WSDL Node

The *WSDL* node is the default node used by the Consume Web Service wizard.

Warning! Do not modify or delete the *WSDL* node.

Prerequisites for Adding and Configuring Nodes

To configure a node and its associated transactions, at least one gateway with one connector must be defined.

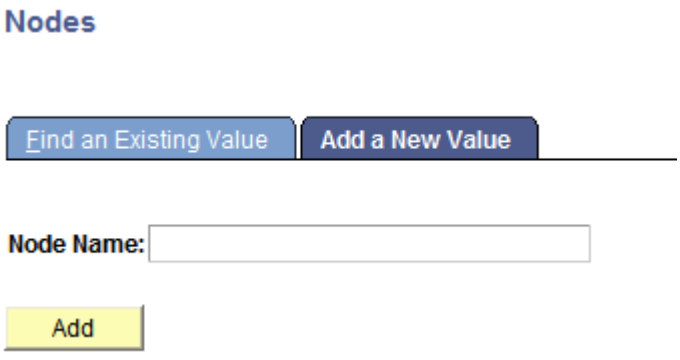
See Chapter 3, "Using the Integration Broker Quick Configuration Page," page 11.

Adding Node Definitions

This section discusses how to add a node definition to the system.

Adding a Node Definition

The following example shows the Nodes-Add a New Value page.



Nodes–Add a New Value page.

Note. The name you specify for a remote node must be the same as the name it specifies for itself.

To add a node:

1. Select PeopleTools, Integration Broker, Integration Setup, Node Definitions.
2. Click the Add a New Value tab.
3. In the Node Name field, enter a name for the node, keeping in mind that node names must begin with a character and may contain up to 30 characters.
4. Click the Add button to define the node.

   The Node Definitions tab displays.

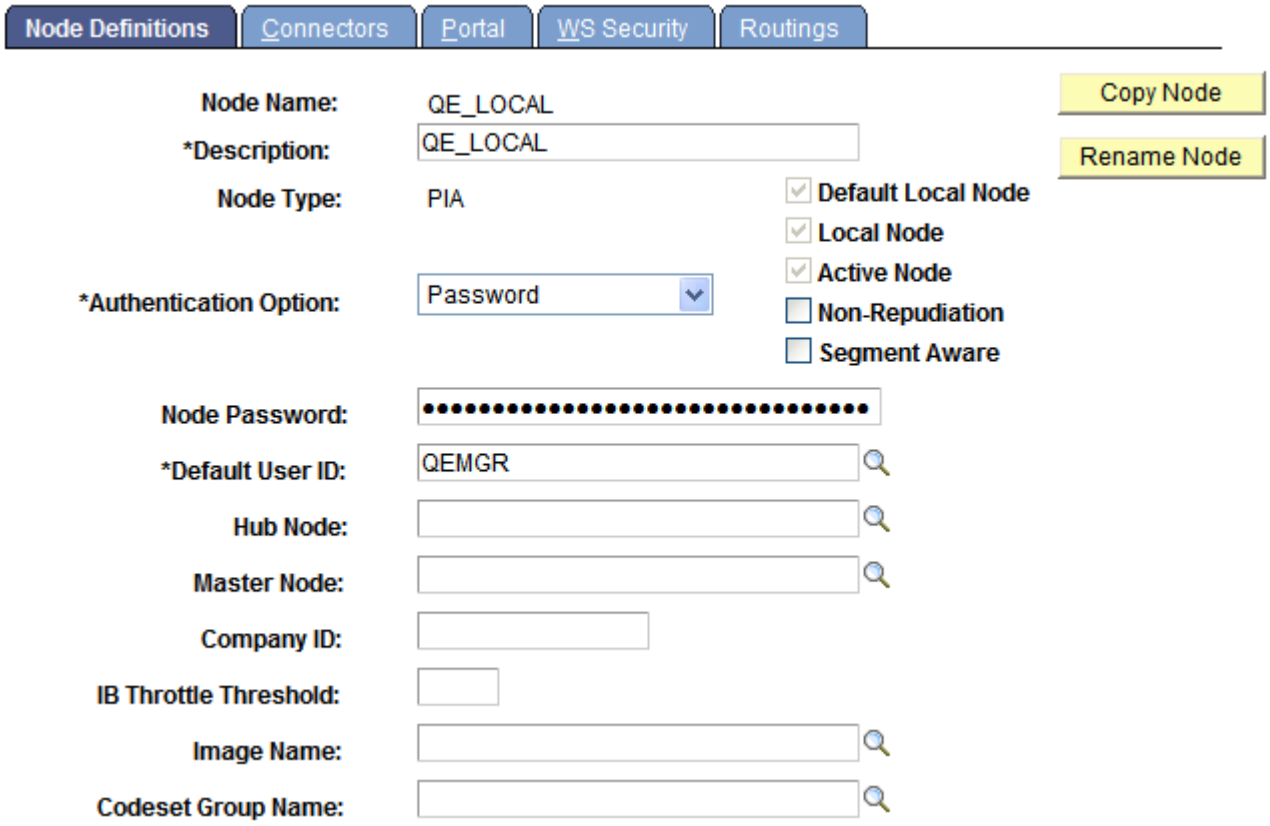
Configuring Nodes

This section discusses how to:

- Define node parameters.
- Specify contact information.
- Define node properties.
- Specify node gateways and connectors.

Defining Node Parameters

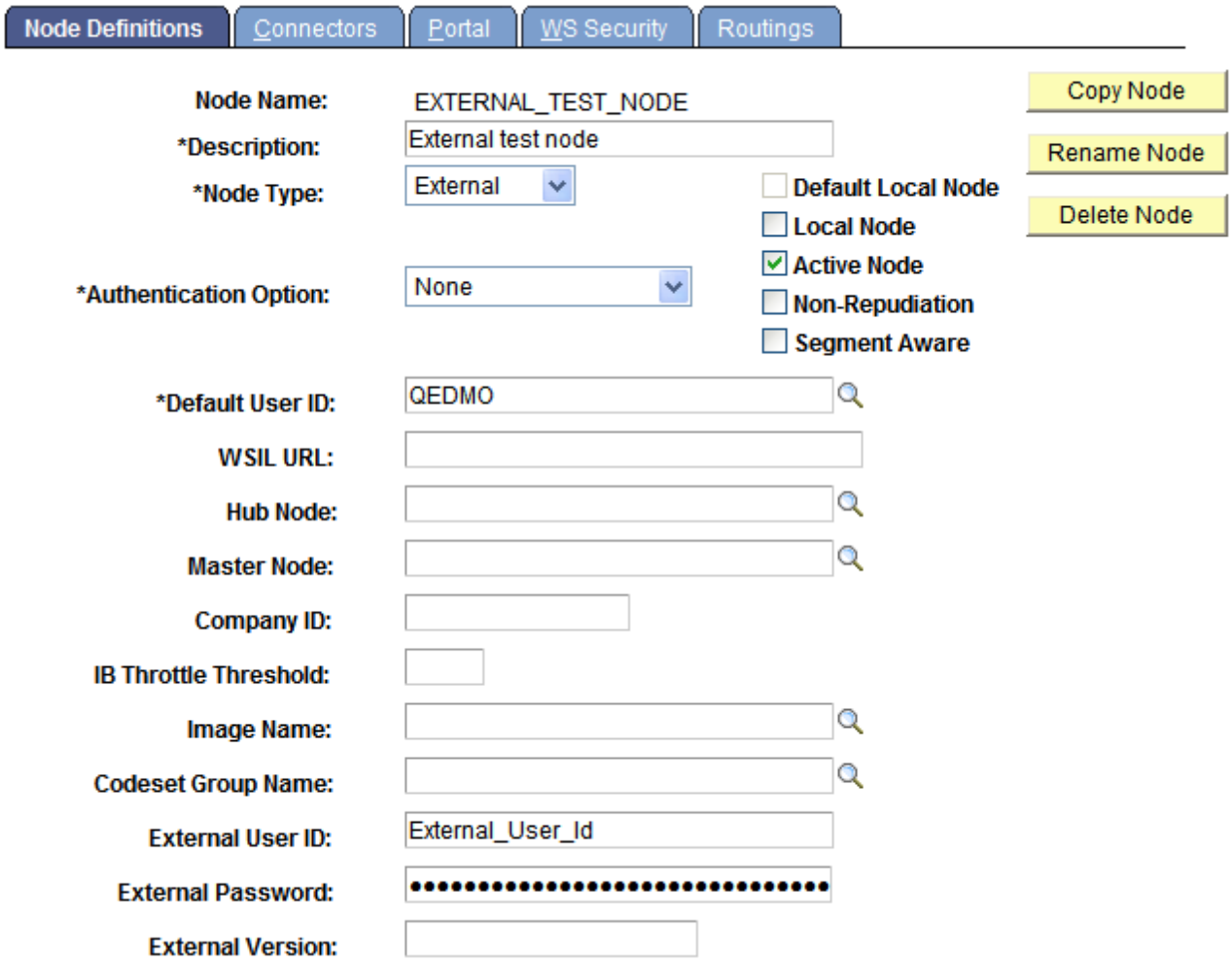
Access the Node Definitions page (PeopleTools, Integration Broker, Integration Setup, Nodes).



Node definition the default local node, QE_LOCAL.

The previous example shows the node definition for the node *QE_LOCAL*, the default local node.

The following example shows the node definition for the node *QE_EXTERNAL*, an *External* type node.



Definition for an external node.

Description Enter a descriptive name for the node.

Node Type Select from:

PIA: Designates the node as a PeopleSoft database that uses PeopleSoft Integration Broker. This is the default for a new node.

External: Designates the node as an entity that doesn't use PeopleSoft Integration Broker.

ICType: A portal-specific setting that PeopleSoft Integration Broker doesn't use.

Authentication Option Select from:

- *Certificate*: The current node uses a digital certificate to sign the messages it sends, and expects messages it receives to be signed by a complementary digital certificate. When a PeopleSoft Pure Internet Architecture node receives a service operation, PeopleSoft Integration Broker extracts the distinguished name from the certificate and validates it against the sending node's distinguished name retrieved from the default local node's keystore. Service operations sent by the default local node have the digital certificate automatically inserted by Integration Broker. An external node is expected to respond to certificates outwardly the same way as a PeopleSoft Pure Internet Architecture node.
- *None:* No authentication is required. This is the default value.

  Warning! Single signon is not compatible with this option. If you select *None* for the default local node, and implement single signon on the same system, all transactions will fail. You must select either *Password* or *Certificate* when implementing single signon.
- *Password*: Two new fields appear: Password and Confirm Password. Enter your password in the first edit box, and confirm it in the second edit box. With a PeopleSoft Pure Internet Architecture node, PeopleSoft Integration Broker expects service operations, both outbound to and inbound from the current node, to include a password, which it validates against the password entered here. An external node is expected to respond to passwords outwardly the same way as a PeopleSoft Pure Internet Architecture node.

See Chapter 11, "Setting Up Secure Integration Environments," Implementing Node Authentication, page 236.

Default Local Node Indicates whether the current node represents the database to which you are assigned.

Local Node Indicates that the current node is either a portal node or the default local node.

You cannot change this setting for the default local node.

Active Node Select to make the current node definition active, so it can be used by PeopleSoft Integration Broker.

Clear the box to inactivate the node.

Note the following points about inactivating a node:

- You cannot inactivate the default local node.
- Inactivating a node will inactivate related routing definitions. You must reactivate the routing definitions manually.

  See *Enterprise PeopleTools 8.50 PeopleBook: PeopleSoft Integration Broker*, "Managing Service Operation Routing Definitions," Activating and Inactivating Routing Definitions.

Non-Repudiation Select to activate nonrepudiation for the current node.

Note that to activate nonrepudiation for the current node you must also activate nonrepudiation in the service operation definition for which you want this feature.

Segment Aware Check the box to configure the node to handle message segments.

See *Enterprise PeopleTools 8.50 PeopleBook: PeopleSoft Integration Broker*, "Sending and Receiving Messages," Working With Message Segments.

Password Displays when the Authentication Option is *Password.*

Enter the node password.

Confirm Password Reenter the node password you entered in the Password field.

Default User ID On inbound integrations, this is the user ID that the sender must specify to invoke a service operation, unless you have set up an external user ID for this purpose.

On outbound integrations, this is the default user ID sent with the service operation.

WSIL URL This field appears only when working with an *External* node type.

This field is used in conjunction with using introspection to create routing definitions.

Enter the WSIL URL for the target system to include in the routing definition.

Hub Node Select the name of a node that will serve as a "gatekeeper" for the current node. You can select any existing PeopleSoft Pure Internet Architecture node for this purpose.

Not all node types are appropriate as hub nodes. Nodes of type ICType are portal-specific, and aren't used by PeopleSoft Integration Broker. A node of type External typically isn't an Integration Broker system, so it might not be usable as a hub node unless you've explicitly configured it to be compatible with Integration Broker.

Master Node This field is for information only. If the current node is used as a hub, you can indicate the target node with which it's associated. If the current node represents a subordinate database, you can indicate the primary database.

Company ID Enter the name of the company or organization associated with the current node.

IB Throttle Threshold Set this parameter on a remote node definition to limit the number of requests sent to the node per dispatch. The setting is in minutes.

For slow-processing systems, this option can help to prevent saturating the targeting system with requests.

This parameter is used only for asynchronous integrations.

Image Name Select an image from the system database. Any application that uses images can use the selected image to represent the current node.

Code Set Group Name Select the codeset group to which you want the current node to belong. Transform programs invoked by service operations use this association to search for message data requiring translation.

See *Enterprise PeopleTools 8.50 PeopleBook: PeopleSoft Integration Broker*, "Applying Filtering, Transformation and Translation," Performing Data Translation.

Copy Node The Copy Node button displays after you have saved the initial node definition.

Click to define a new node with the same properties as the current node. The Default Local check box is cleared for all new nodes.

Note. If you copy a local node, the new node will be local as well. You must clear the Local Node check box to use it with PeopleSoft Integration Broker.

Rename Node The Rename Node button displays after you have saved the initial node definition.

You can rename only the default local node.

Additional information about deleting nodes is contained elsewhere in this chapter.

See Chapter 7, "Adding and Configuring Nodes," Renaming or Deleting Nodes, page 139.

Delete Node The Delete Node button displays after you have saved the initial node definition.

You cannot delete the default local node.

Additional information about deleting nodes is contained elsewhere in this chapter.

See Chapter 7, "Adding and Configuring Nodes," Renaming or Deleting Nodes, page 139.

External ID This field appears only when working with an *External* node type.

This field is used for outbound integrations in conjunction with implementing WS-security.

See Chapter 11, "Setting Up Secure Integration Environments," Specifying External User IDs and Passwords, page 204.

External Password This field appears only when working with an *External* node type.

This field is used for outbound integrations in conjunction with implementing WS-security.

See Chapter 11, "Setting Up Secure Integration Environments," Specifying External User IDs and Passwords, page 204.

External Version This field appears only when working with an *External* node type.

This field is currently not used.

Specifying Contact Information

Click the Contacts/Notes link to access the Node−Node Contacts/Notes page.

Each node represents a database or other software entity managed by one or more people. Use this page to record information about the people associated with the current node.

Contact Manager The name of the representative or administrator of the node, in standard PeopleSoft name format.

Contact Email The Contact Manager's email address, in standard PeopleSoft email address format.

Contact Phone Number The phone number of the contact manager.

Contact URL The address of the Contact Manager's support web site, if there is one.

Defining Node Properties

Click the Properties link to access the Node − Properties page.

This page provides a convenient place to store additional information about the current node that can be referenced by any other node. Properties created for all nodes are stored in a single table, PSNODEPROP.

Examples include a DUNS number or Tax Identification Number. These properties can be used to update messages with additional information. They can also serve to add additional categorization for custom processing; for example, add a Region property so nodes can be referenced by region for special processing.

Name Type Select from:

Category: The property is used for categorization.

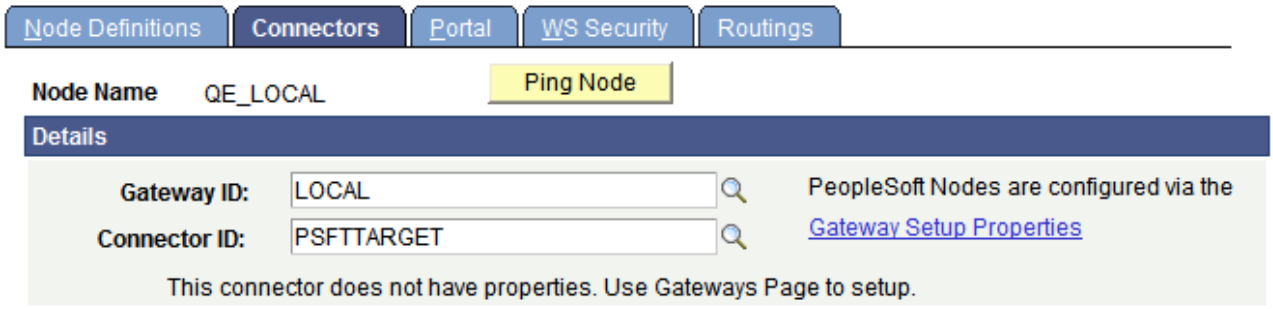
Ident: The property is used for identification.

Search: The property is used for searching.

Property Name Enter a new property name or select an existing property of the selected name type.

Specifying Gateways and Connectors

Select the Connectors tab to access the Node − Connectors page.



Nodes-Connectors page

Use this page to specify the integration gateway and target connector the node uses for integrations.

The default target connector is the PeopleSoft target connector (PSFTTARGET). There are no default or required properties associated with this connector, so no properties grid appears when you first access this page or if you continue to use this connector.

At least one gateway with at least one target connector must be defined and configured. If the current node is remote, it can use the default local node's gateway or any other installed gateway as its local gateway. If the current node has its own gateway installed, the default local node's database must contain a definition for it, configured as a remote gateway.

Specifying a Gateway and Target Connector for the Current Node

To specify a gateway and connector for the current node:

1. From the Gateway ID field, click the Lookup button to select the gateway ID for the gateway you want the current node to use.

   When the default local node sends a message to any other node, the message first goes to the default local node's local gateway through its PeopleSoft listening connector, regardless of the gateway ID you select here.

   - If you specify a remote gateway ID, the local gateway uses its default remote gateway connector (specified in the integrationGateway.properties file) to route messages to the remote gateway through the remote gateway's PeopleSoft listening connector. The remote gateway sends the messages directly to the current node, using the connector you specify in the next step.

     Note. The default remote gateway connector setting initially specifies the HTTP target connector, which is unlikely to change unless you develop a custom target connector.

   - If you specify the local gateway ID, the local gateway sends messages directly to the current node, using the connector you specify in the next step.

2. From the Connector ID field, select a connector ID from the list of connectors registered with the selected gateway.

   Specify the target connector appropriate to the communication method preferred by the current node. If the node is a PeopleSoft application with Integration Broker installed, select *PSFTTARGET*. If the node is a PeopleSoft 8.1x application, select *PSFT81TARGET*.

   The rows on the Properties and Data Type/Description tabs are automatically populated with the connector's properties that are designated *Required* in the gateway definition. The fields on these tabs are the same as those on the Connector Properties page. If the connector has multiple instances of a required property defined, only the instance designated as *Default* appears.

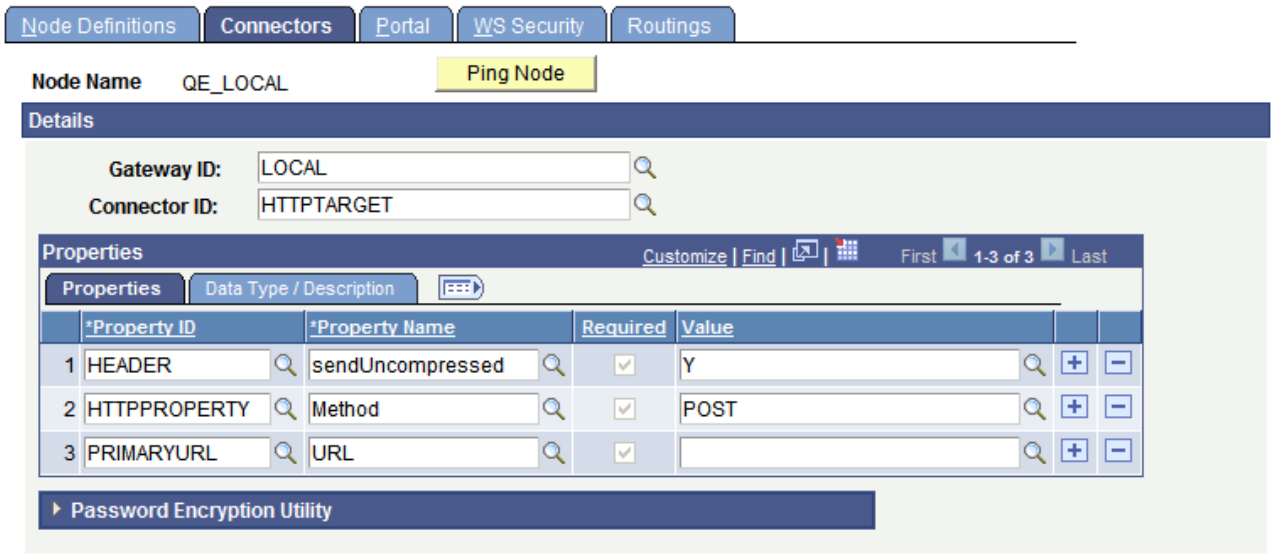
   See Chapter 5, "Managing Integration Gateways," Editing Connector Properties, page 36.

3. Click the Save button.

Note. You can override the gateway and connector selection for individual outbound transactions.

Working With Connector Properties

Properties that appear on this page are copies of the specified connector's required properties. The following example shows the node definition configured with the HTTP target connector (HTTPTARGET). The default properties for this connector appear in the Properties grid.



HTTP target connector properties

You can use this page to:

- Add an instance of a non-required property.
- Add a new instance of a required property.
- Modify the value or description of a property instance.
- Remove a property instance.

Information about appropriate modifications might come from PeopleSoft, from the connector's developer, or from your own experience and requirements.

Important! Don't remove a required property unless you replace it with another instance of the same property. Without all of its required properties, the connector is unlikely to work correctly.

You must encrypt any password connector property values. The Connector tab features access to the Password Encryption Utility that enables you to encrypt a password value and paste it into the appropriate field on the page. To access the utility, click the Password Encryption Utility arrow.

Accessing the Integration Gateway Properties File

The Connectors tab features a Gateway Setup Properties link you can use to access the integrationGateway.properties file directly from this tab.

Testing Connector Configurations

The Connectors tab features a Ping Node button you can use to test your configuration.

If the ping is successful, a window displays with a message indicating that the gateway is active and the PeopleTools version that you are running. If the ping is not successful, a window displays with a message indicating the gateway could not be found.

See Also

Chapter 5, "Managing Integration Gateways," Editing Connector Properties, page 36

Renaming or Deleting Nodes

This section discusses how to rename and delete nodes.

Understanding Renaming and Deleting Nodes

This section discusses how to rename and delete nodes.

There are several situations in which you might need to rename or delete a node definition. When you do so, PeopleSoft Integration Broker automatically handles most of the dependencies involved — such as deleting routings and other properties associated with the node.

However, the live message data in Integration Broker Monitor remains unchanged. If that data still contains references to the node you want to modify, Integration Broker will prevent you from making the modification. You must remove all data from the live message tables before you can rename or delete the node definition.

You cannot delete the default local node or a node that hosts a portal. As a result, the Delete Node button is hidden on these node definitions.

Note. If you upgraded your PeopleSoft application from a PeopleTools 8.1x release, the newly created default local node definition must be renamed, so you must first remove any remaining live message data if you didn't do so before the upgrade.

Renaming or Deleting a Node

Renaming or deleting a node requires the following actions:

1. Deactivate all the domains in your messaging system.
   a. Access the Domain Status page.
   b. For each active domain in the system, from the Domain drop-down list box, select Inactive.
   c. Click Update to change the status of all domains to *Inactive* and all dispatchers to *Cleanup.*
   d. Click Force Reset to change the status of all dispatchers to *Inactive.*
2. Remove the data from the live message tables.

   You have several choices when removing data from the live message tables:

   - You can archive messages one at a time from the Asynchronous Details or Synchronous Details component.
   - You can archive messages with a batch process using the Archive Monitor Data component.
   - You can purge message data using one of several Data Mover scripts delivered with PeopleSoft Integration Broker. You'll find them in *PS_HOME*\scripts:

     AppMsgPurgeLive.dms Deletes the queue data from every live message table in the database.

     AppMsgPurgeAll.dms Deletes the message data from every live message table and every archive message table in the database. This is the recommended procedure when upgrading from earlier versions of PeopleTools, because the archived data is largely incompatible with the new release.

3. Rename or delete the desired node definition.

   If you are renaming the default local node, note that the name cannot exceed 15 characters. Other node names can contain up to 30 characters.
4. Reboot the web server.
5. Reactivate the messaging domains.
   a. Access the Domain status page.
   b. On the Domain Status page, select All Domains Active.
   c. Click Update to change the status of all domains and dispatchers to *Active.*

See Also

Enterprise PeopleTools 8.50 PeopleBook: Integration Broker Service Operations Monitor, "Running Batch Service Operation Archiving Processes"

Enterprise PeopleTools 8.50 PeopleBook: Integration Broker Service Operations Monitor, "Purging Runtime Service Operations Monitor Tables"

Enterprise PeopleTools 8.50 PeopleBook: Data Management, "Using PeopleSoft Data Mover," Understanding Data Mover Scripts

Chapter 8

Configuring PeopleSoft Integration Broker for Handling Services

This section discusses how to configure PeopleSoft Integration Broker for handling services.

Understanding Configuring PeopleSoft Integration Broker for Handling Services

This section provides an overview of several of the service configuration properties that you must set to use services with PeopleSoft Integration Broker.

Namespaces

Namespaces provide a method for qualifying element and attribute names that are used in XML documents and are identified by Uniform Resource Identifier (URI) references.

To use services with PeopleSoft Integration Broker, you must specify a service namespace and a schema namespace.

Target Locations

Target locations are URLs that PeopleSoft Integration Broker uses to build and validate XML message schemas, export WSDL documents, and as the SOAP endpoint.

PeopleSoft Integration Broker enables you to define two target location URLs, and provides a Target Location field and Secure Target Location field on the Service Configuration page.

The URL you enter in the Target Location field should be an unsecured URL. By default, the system uses this URL to build and validate XML message schemas, export WSDL documents, and as the SOAP endpoint.

Note. XML schema validation will fail if you enter a secured URL in the Target Location field.

If you need to use a secure URL for the SOAP endpoint, enter a secure URL in the Secure Target Location field. Any URL you enter in the Secure Target Location field overrides the URL entered in the Target Location field for the SOAP endpoint.

In addition, you can override the (unsecured) target location URL for exporting WSDL documents on a case–by–case basis. The Provide Web Service wizard features a Use Secure Target Location box in Step 2 – Select Service Operations of the Wizard. If you select the box, the system exports the WSDL document to the URL specified in the Secure Target Location field on the Service Configuration page. The override is in effect only for that particular export of WSDL. If you need to generate WSDL again for the service, the system defaults back to using the (unsecured) target location URL, unless you again check the Use Secure Target Location box in the wizard to override the URL to use the secure target location.

To use services with PeopleSoft Integration Broker, you must specify a target location. Specifying a secure target location is optional.

Service System Status

The Services Configuration page contains a Service System Status drop-down list box that enables you to restrict rename, delete, and other administrative actions that users can perform on services, service operations, messages, and other integration metadata.

You can select one of two values from the drop-down list box: *Production* or *Development.*

The following table describes the impact of the setting on managing integration metadata:

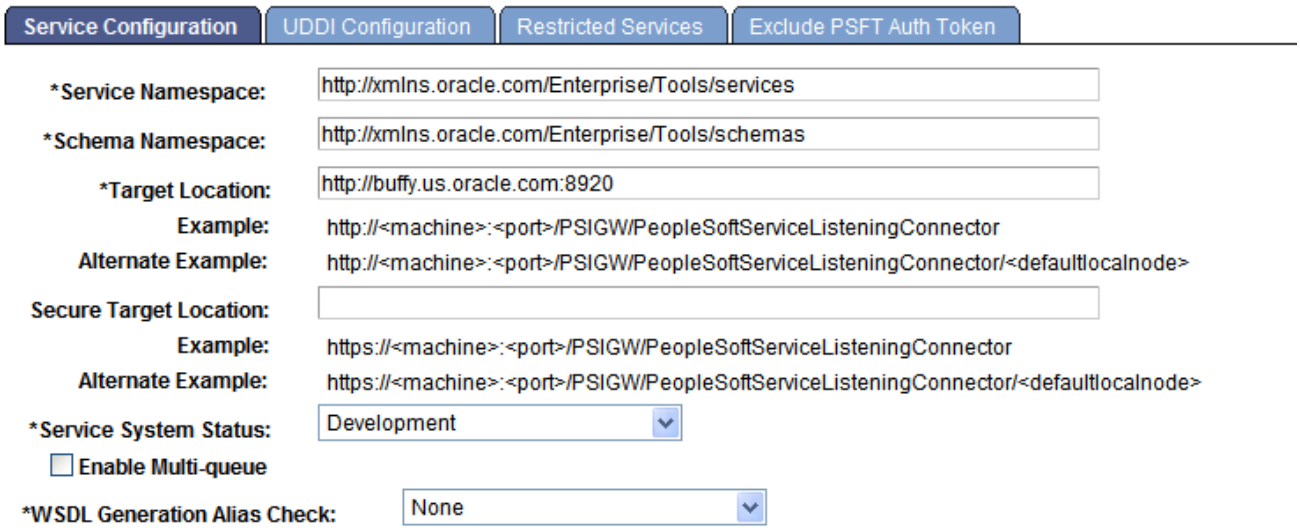
Object	***Action***	***Production Mode***	***Development Mode***
Messages	Rename	You cannot rename a message that is associated to a service that has WSDL provided. You must first delete the WSDL documents before you can rename the message.	An alert message displays indicating that WSDL documents have been provided for the service to which the message is associated, but you may continue with the action and rename the message.
Messages	Delete	You cannot delete a message that is associated to a service that has WSDL provided. You must first delete the WSDL documents before you can delete the message.	An alert message displays indicating that WSDL documents have been provided for the service to which the message is associated, but you may continue with the action and delete the message.
Message Schemas	Delete	You cannot delete a message schemas that is associated to a service that has WSDL provided. You must first delete any WSDL documents before you can rename the schema.	An alert message displays indicating that WSDL documents have been provided for the service to which the message schema is associated, but you may continue with the action and rename the schema
Queues	Rename	The Service System Status has no impact on renaming queue definitions. However, you cannot rename a queue if it is referenced in a service operation or if it is referenced in the runtime tables.	The information that applies to renaming queues in production mode also applies to renaming queues in development mode.
Queues	Delete	The Service System Status has no impact on deleting queue definitions. However, you cannot delete a queue if it is referenced in a service operations or if it is referenced in a runtime table.	The information that applies to deleting queues in production mode also applies to deleting queues in development mode.

Object	*Action*	*Production Mode*	*Development Mode*
Routings	Rename	The Service System Status has no impact on renaming routing definitions.	The information that applies to renaming routing definitions in production mode also applies to renaming routings in development mode.
Routings	Delete	You cannot delete an any-to-local routing definition that is tied to a service that has WSDL provided. You must first delete the WSDL from the service before deleting the routing definition.	An alert message displays indicating that WSDL documents have been provided for the service to which the routing is associated, but you may continue with the action and delete the routing definition.
Service	Rename	You cannot rename services that have had WSDL documents provided. The WSDL documents must be deleted before you can rename a service.	An alert message displays indicating that WSDL documents have been provided for the service, but you can continue with the action and rename the service.
Service	Delete	The Service System Status has no impact on deleting services. However, you cannot delete any service that is referenced by a service operation.	The information that applies to deleting services in production mode also applies to deleting services in development mode.
Service Operation	Rename	You cannot rename service operations that are associated to services that have WSDL provided. You must delete the WSDL before you can rename the service operation.	An alert message displays indicating that WSDL documents have been provided for the associated service, but you may continue with the action and rename the service operation.
Service Operation	Delete	• You cannot delete service operations that are associated to services that have WSDL provided. You must delete the WSDL before you can delete the service operation. • If you delete the default service operation version, all versions of the service operation are deleted. You cannot delete a service operation that is referenced in the runtime tables.	An alert message displays indicating that WSDL documents have been provided for the associated service, but you may continue with the action and delete the service operation. You cannot delete a service operation that is referenced in the runtime tables.
Service Operation	Change Service	You cannot change a service operation that is associated to a service that has WSDL provided. You must first delete the WSDL documents before you can modify the setting. The new service to which you associate an operation may have had WSDL generated. However, if you want the WSDL from the newly associated service operation to be included in the WSDL document, you must export the service again.	An alert message displays indicating that WSDL documents have been provided for the associated service, but you may continue with the action and change the service associated with the service operation. The new service to which you associate an operation may have had WSDL generated. However, if you want the WSDL from the newly associated service operation to be included in the WSDL document, you must export the service again.

Setting Service Configuration Properties

This section discusses how to set service configuration properties.

You set service configuration properties on the Service Configuration page (IB_SVCSETUP) in the Services Configuration component (IB_SVCSETUP). To access this page, select PeopleTools, Integration Broker, Services Configuration. The following example shows the Service Configuration page:



Service Configuration page

To set service configuration properties:

1. Access the Service Configuration Properties page (PeopleTools, Integration Broker, Services Configuration).
2. In the Service Namespace field, declare a service namespace.
3. In the Schema Namespace field, declare a schema namespace.
4. In the Target Location field, enter an unsecured URL to be used for XML message schema, WSDL, and as the SOAP endpoint.

   If you enter a secure URL in the Target Location field, XML message schema validation will fail.

   If you require a secure target location for the SOAP endpoint, see Step 5.

   If you have a dedicated integration gateway, the format of the value that you enter is http://<machine>:<port>/PSIGW/PeopleSoftServiceListeningConnector.

   If the default local node points to a different gateway server where WSDL documents and XSD schemas are available, use the alternate location URL format of http://<machine>/PSIGW/PeopleSoftServiceListeningConnector/<defaultnode>.

5. (Optional) In the Secure Target Location field, enter a secure URL to be used as the SOAP endpoint.

   The URL entered here overrides the target location defined in Step 4 for the target location for the SOAP endpoint.

   The URL you enter here is also used as a secure target location for exporting WSDL, if you choose the Use Secure Target Location box in the Provide Web Service wizard.

   If you do not enter a value in this field, the system uses the unsecured URL that you specify in the Target Location field for both the SOAP endpoint and for exporting WSDL.

   See *Enterprise PeopleTools 8.50 PeopleBook: PeopleSoft Integration Broker*, "Providing Services," Step 2: Select Service Operations.

   If you have a dedicated integration gateway, the format of the value that you enter is http://<machine>:<port>/PSIGW/PeopleSoftServiceListeningConnector.

   If the default local node points to a different gateway server, use the alternate location URL format of http://<machine>/PSIGW/PeopleSoftServiceListeningConnector/<defaultnode>.

6. From the Service System Status drop-down list box, select one of the following options:

   - *Development.* (Default.)
   - *Production.*

   These statuses are discussed elsewhere in this chapter

   See Chapter 8, "Configuring PeopleSoft Integration Broker for Handling Services," Understanding Configuring PeopleSoft Integration Broker for Handling Services, page 143.

7. (Optional) Check the Enable Multi-queue box to use multiple queues to process inbound and outbound asynchronous requests.

   This feature is discussed elsewhere in PeopleBooks.

   See *Enterprise PeopleTools 8.50 PeopleBook: PeopleSoft Integration Broker*, "Managing Service Operations," Assigning Multiple Queues to Process Service Operations.

8. From the WSDL Generation Alias Check drop-down list, select an option for enforcing field and record alias names in generated WSDL.

   This feature is discussed elsewhere in PeopleBooks.

   See *Enterprise PeopleTools 8.50 PeopleBook: PeopleSoft Integration Broker*, "Managing Messages," Enforcing Message Record and Field Aliases in Generated WSDL.

9. Click the Save button.

See Also

Chapter 8, "Configuring PeopleSoft Integration Broker for Handling Services," Target Locations, page 143

Chapter 8, "Configuring PeopleSoft Integration Broker for Handling Services," Namespaces, page 143

Chapter 5, "Managing Integration Gateways," Setting the Namespace for Generic SOAP Faults, page 53

Chapter 9

Specifying UDDI Repositories in PeopleSoft Systems for Providing and Consuming Services

This chapter discusses how to specify UDDI repositories in PeopleSoft systems for providing and consuming services.

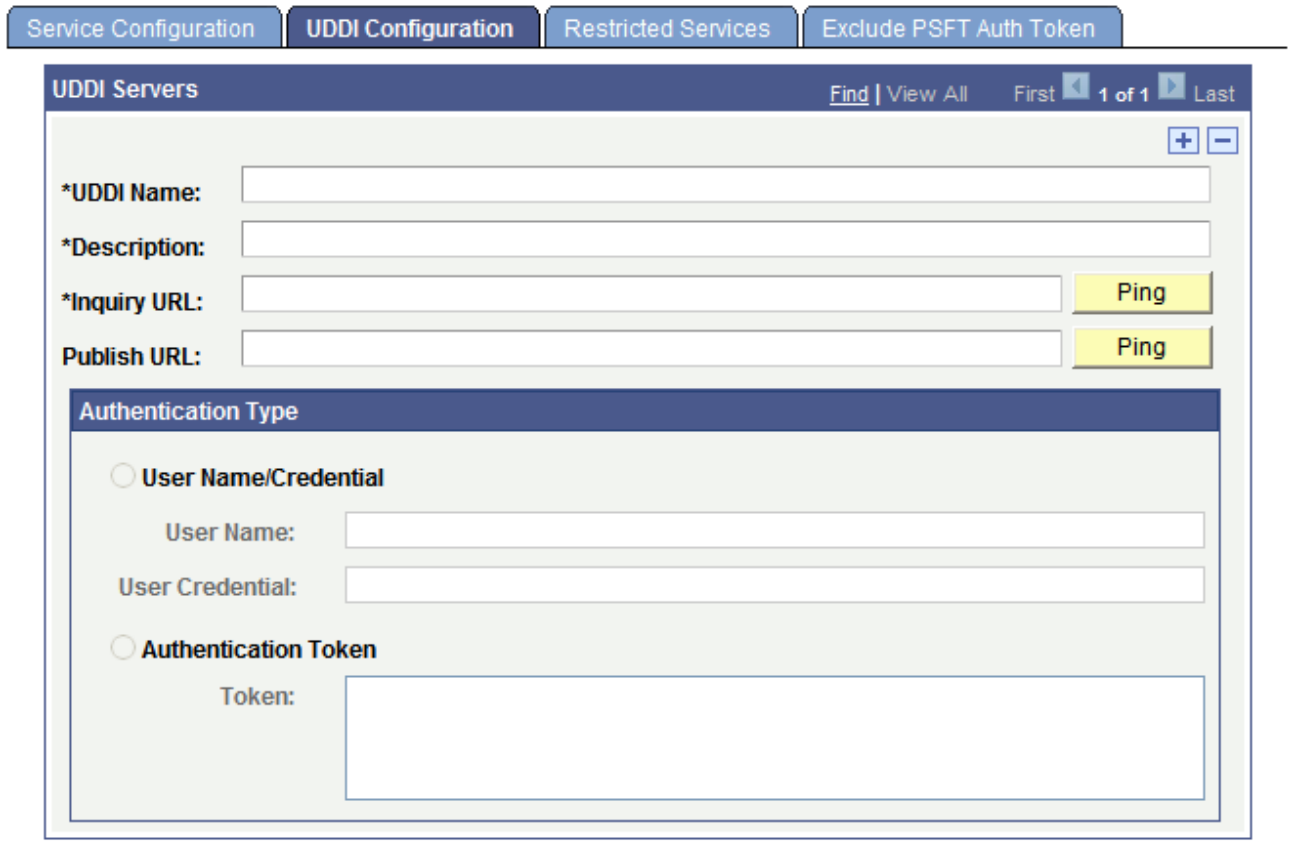
Understanding Specifying UDDI Repositories in PeopleSoft Systems

You can provide services to and consume services from one or more UDDI repositories. Before doing so, you must configure each repository in the PeopleSoft system.

Specifying UDDI Repositories in the PeopleSoft System

Use the Service Configuration-UDDI Configuration page (IB_SVCSETUP2) to specify UDDI repositories in the PeopleSoft system for providing services to and consuming services from UDDI repositories.

To access this page, select PeopleTools, Integration Broker, Integration Setup, Service Configuration and click the UDDI Configuration tab. The following graphic shows the UDDI Configuration page:



Services Configuration — UDDI Configuration page

To specify a UDDI repository in the PeopleSoft system:

1. Access the UDDI Configuration page (PeopleTools, Integration Broker, Configuration, Services Configuration. Click the UDDI Configuration tab).
2. In the UDDI Name field, enter the name of the UDDI server.
3. In the Description field, enter a descriptive information for the UDDI server.
4. Specify the URL used when consuming services from UDDI repositories.
    a. In the Inquiry URL field, enter the URL to use to inquire for services available on the UDDI server. This is the URL used when consuming services from UDDI repositories. It is also used when publishing to UDDI repositories to inquire the server for possible existing WSDL document versions.
    b. Click the Ping button next to the Inquiry URL field to verify that you entered the correct URL.
5. Specify the Publish URL.
    a. In the Publish URL field, enter the URL for publishing WSDL documents to the UDDI server. This URL is used when providing services to UDDI repositories.
    b. Click the Ping button next to the Publish URL field to verify that you entered the correct URL.

To specify additional UDDI repositories to use for providing or consuming services, click the plus (+) button at the top right corner of the UDDI Server section to add a row.

Chapter 10

Managing Pub/Sub Server Domains

This chapter discusses how to:

- View dispatcher status.
- Activating pub/sub server domains.
- Inactivate pub/sub server domains.
- Change dispatcher status for processes.
- Set domain grace periods.

Understanding Managing Pub/Sub Domains

PeopleSoft Integration Broker includes a set of Oracle Tuxedo servers that monitor database tables and process items in the tables. The processing can include running PeopleCode programs, creating publication and subscription contracts, and so forth.

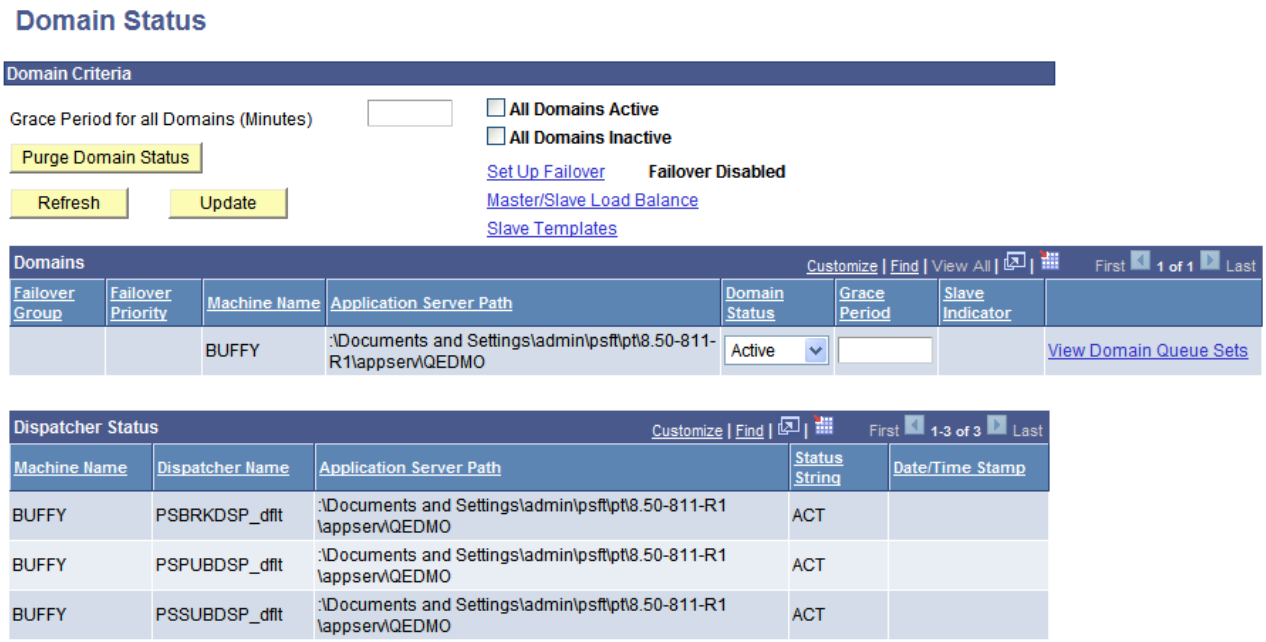
The Domain Status page enables you to view the domains that have pub/sub servers on them that are running against the application database. You can also use this page to manually set domain grace periods to allow processing in a domain to finish before you pause the processing or take the domain offline.

In addition, if a machine with a domain on it crashes, the integration system may still operate as if the processes in the domain are still working on items in the runtime tables. The Domain Status page enables you to set the domains to inactive so that other pub/sub servers can complete the processing of these items. This enables you to recover from domain and machine crashes.

Working with the Domain Status Page

The Domain Status page (AMM_MULTIDOM) features three sections, the Domain Criteria section, the Domain Status section, and the Dispatcher Status section.

The following example shows the Domain Status page:



Domain Status page

The Domain Criteria section enables you to perform actions on all domains in the integration system, such as apply a grace period to all domains, activate or inactivate all domains, and purge the current information in the Dispatcher Status section.

The Domains section enables you to activate and inactivate domain status and set domain grace periods. You can also use this section to view failover information for a domain.

The Domain Status section provides application server name and path information for all machines that have domains on the messaging system. For any machine, you can use the drop-down list box to activate or inactivate the machine and all domains on it. You can also set grace periods for domains on specific machines.

The Domain Status page also features the following controls:

Purge Domain Status Click to purge all of the current status information in the Dispatcher Status section. After you click this button, the system populates the section with information about all processes that are still running.

Update Click to saves or apply changes that you make in the Domain Criteria section or the Domain Status section.

Force Reset Click to reset the status of all entries in the Dispatcher Status column in the Dispatcher Status section to *Inactive*.

Refresh Click to refresh the Domains section and Dispatcher Status section of the page.

Viewing Dispatcher Status

The Dispatcher Status section of the Domain Status page displays information about machines in the integration system that have dispatcher processes associated with them. This area displays the machine name, the dispatcher process name, the application server path, the dispatcher status, and any grace periods set for a process running on the domain.

There are three valid dispatcher status values:

ACT	Indicates that the dispatcher process is active on the domain.
INACT	Indicates that the dispatcher process is inactive on the domain. No processing occurs.
CLNUP	Indicates that the dispatcher process is in clean-up mode. The pub/sub server releases queued items for processing and waits for items currently processing to finish. The time that appears in the grace period column indicates when the cleanup process will end. The time equals the system time and the clean up time interval that you enter.

Activating Pub/Sub Server Domains

Before you can use the pub/sub system, you must activate the domain on which a pub/sub server resides.

To activate a domain:

1. Select PeopleTools, Integration Broker, Service Operations Monitor, Administration, Domain Status..

   The Domain Status page appears.

2. In the Domains section:

   a. Locate the row that lists the machine where the domain resides that you want to activate.

   b. In the Domain Status drop-down list box, select *Active.*

3. Click the Update button.

Inactivating Pub/Sub Server Domains

To inactivate pub/sub servers on domains:

1. Inactivate pub/sub server domains:

    a. To inactivate domains on all machines in the messaging system, select the All Domains Inactive check box. To activate the servers at a later time, select the All Domains Active box.

    b. To inactivate domains on individual machines, locate the domains to inactivate. In the drop-down list box, select *Inactivate.* To activate the servers at a later time, select *Activate* in the list.

2. Click the Update button.

The domain status for the domains that you inactivate changes from *Active* to *Inactive.* In addition, in the Dispatcher Status section, the dispatcher status of all processes associated with the domains changes from active *(ACT)* to cleanup *(CLNUP).* Click the Refresh button until the dispatcher status changes to inactive *(INACT).*

If you inactivated all domains, a Force Reset button appears under the Update button. The Force Reset button enables you to force the dispatcher status to change from cleanup to inactive.

Changing Dispatcher Status for Processes

The Force Reset button appears only when you change the domain status for all domains on all machines by selecting the All Domains Inactive check box.

To change dispatcher status for all processes on all machines from cleanup to inactive:

1. Click the Force Reset button.

2. Click Update.

Setting Domain Grace Periods

The time that appears in the Grace Period column indicates when the cleanup process ends. The time equals the system time and the cleanup time interval that you enter.

To set one grace period to apply to domains on all machines, locate the Grace Period for all Domains field in the Domain Criteria section and enter the number of minutes for the grace period. Click Update.

To set grace periods for individual domains, enter the number of minutes for the grace period for each domain. Click Update.

A grace period that you set for an individual domain takes precedence over the setting for all groups.

The grace period setting for all domains is a convenient way to set a grace period for all dispatchers in all the domains. You can set a grace period of all domains at the top of the page and then press the Tab key to access individual domains and override the group setting.

Chapter 11

Setting Up Secure Integration Environments

This chapter provides an overview of securing integration environments, outbound PeopleSoft Integration Broker security processing, and outbound PeopleSoft Integration Broker security processing, and discusses how to:

- Install application server-based digital certificates.
- Install integration gateway-based digital certificates.
- Install web server-based digital certificates.
- Implement web server SSL encryption.
- Implement WS-Security.
- Implement client authentication.
- Implement nonrepudiation.
- Manage user authentication.
- Implement node authentication.
- Secure service operations with permission lists.
- Validate security on inbound service operations.

Understanding Securing Integration Environments

This section discusses types of integration security and provides an overview of security terminology used in conjunction with PeopleSoft Integration Broker.

Web Server SSL Encryption

Encryption supports data privacy. When encryption is implemented, the sender translates the content of a transaction into a secret code that only the receiver can decrypt. PeopleSoft Integration Broker employs the Secure Sockets Layer (SSL) protocol for data encryption.

You can employ SSL encryption at the web server level to secure data sent between web servers.

You can implement web server SSL to encrypt data sent between your web server and that of your integration partners.

You can implement web server SSL encryption with integration partners running on all PeopleTools 8.4x systems and third-party systems.

You use digital certificates to implement SSL encryption.

WS-Security

Web services security (WS-Security) is implemented on the integration gateway for inbound and outbound integrations with third-party systems.

You can implement WS-Security using username tokens or Security Assertion Markup Language (SAML) tokens .

You can implement WS-Security with integration partners running on PeopleTools 8.48 and later systems and third-party systems.

WS-Security using Username Token Profile

The WS-Security Username Token Profile defines a standard way of identifying the requestor by "username", and optionally using a password (or shared secret, or password equivalent) to authenticate that identity to the web service producer.

On outbound request processing, PeopleSoft Integration Broker generates a WS-Security UsernameToken, which may include a password. The WS-Security information is added to the SOAP request on the integration gateway prior to sending to the integration partner.

On inbound processing, PeopleSoft Integration Broker can process requests received from integration partners that contain WS-Security UsernameToken and password in the SOAP header of the inbound SOAP request.

WS-Security using SAML Token Profile

The SAML Token Profile uses assertions to define a standard way to associate common information such as issuer ID, assertion ID, subject and so on.

On outbound request processing, PeopleSoft Integration Broker adds a WS-Security SOAP header to the service operation that contains SAML credentials defined in the node definition for the node.

On inbound processing, the PeopleSoft system checks for the existence of a WS-Security SOAP header. If it exists, the integration gateway decrypts the SAML token (if it has been encrypted) to restore the user ID information to clear text format.

See Also

Enterprise PeopleTools 8.50 PeopleBook: Security Administration, "Working with Web Service Security (WS-Security)"

Client Authentication

Outbound requests connect from the application server to the integration gateway using an MIME over HTTP connection. To secure the connection you can employ client authentication. This option is typically implemented when the application server and integration gateway reside on separate machines. Client authentication is used only on outbound transactions, since inbound transactions connect between the integration gateway and application server are made using Jolt connection strings.

Note. If you implement client authentication you must also implement web server SSL encryption.

You can implement client authentication with integration partners running on all PeopleSoft 8.4x systems and third-party systems.

Nonrepudiation

Nonrepudiation is a form a digital security that ensures that a transferred message has been sent and received by the parties claiming to have sent and received the message. It is also a method of guaranteeing that the sender of a message cannot later deny having sent the message and that the recipient cannot deny having received the message.

You can implement nonrepudiation with integration partners running on all PeopleSoft 8.4x systems and third-party systems.

User Authentication

Service operations are secured at the user level. On an outbound transaction, user authentication sets the user ID to assign to the service operation.

When user authentication is implemented a user ID or user ID and password are required.

For inbound transactions, user authentication determines the user ID associated with the inbound service operation. If a user ID and password are required to invoke a service operation, the system validates the user ID to see if it is a member of the permission list to which the service operation is assigned.

You can implement user authentication with integration partners running on PeopleSoft 8.48 and later systems and third-party systems.

Node Authentication

Use node-level security for integrations with nodes running on earlier PeopleTools 8.4x releases.

To implement node-level security you define an authentication option for the node using the Nodes page. You can use a node certificate or a password as authentication options.

Node-level security pertains to inbound and outbound processing and authentication is performed on the application server.

You can implement node authentication with integration partners running on all PeopleSoft 8.4x systems and third-party systems.

Service Operation Permission Lists

The user ID that is authenticated during user authentication is validated against the permission list to which the service operation is assigned.

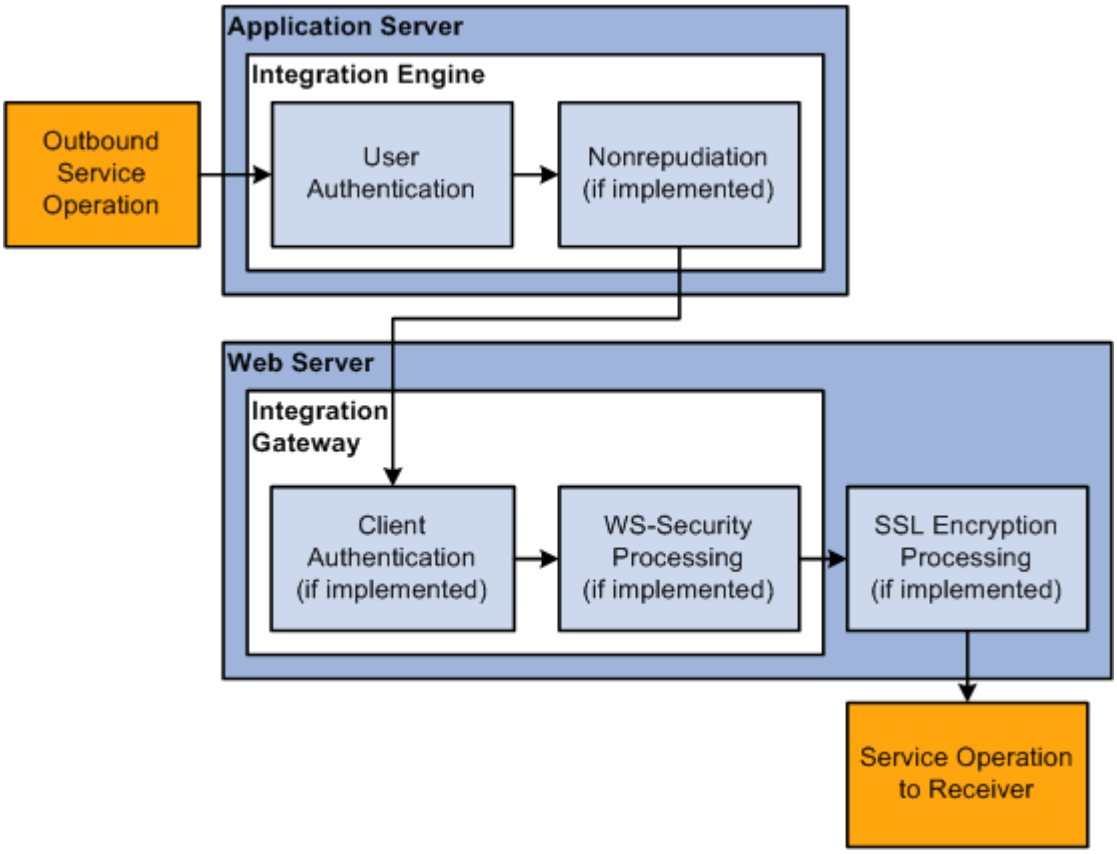
Understanding PeopleSoft Integration Broker Security Processing

This section discusses:

- Outbound PeopleSoft Integration Broker security processing.
- Inbound PeopleSoft Integration Broker security processing.

Outbound Integration Broker Security Processing

The following diagram illustrates security processing for outbound integrations from PeopleSoft Integration Broker:



Outbound PeopleSoft Integration Broker Security Processing

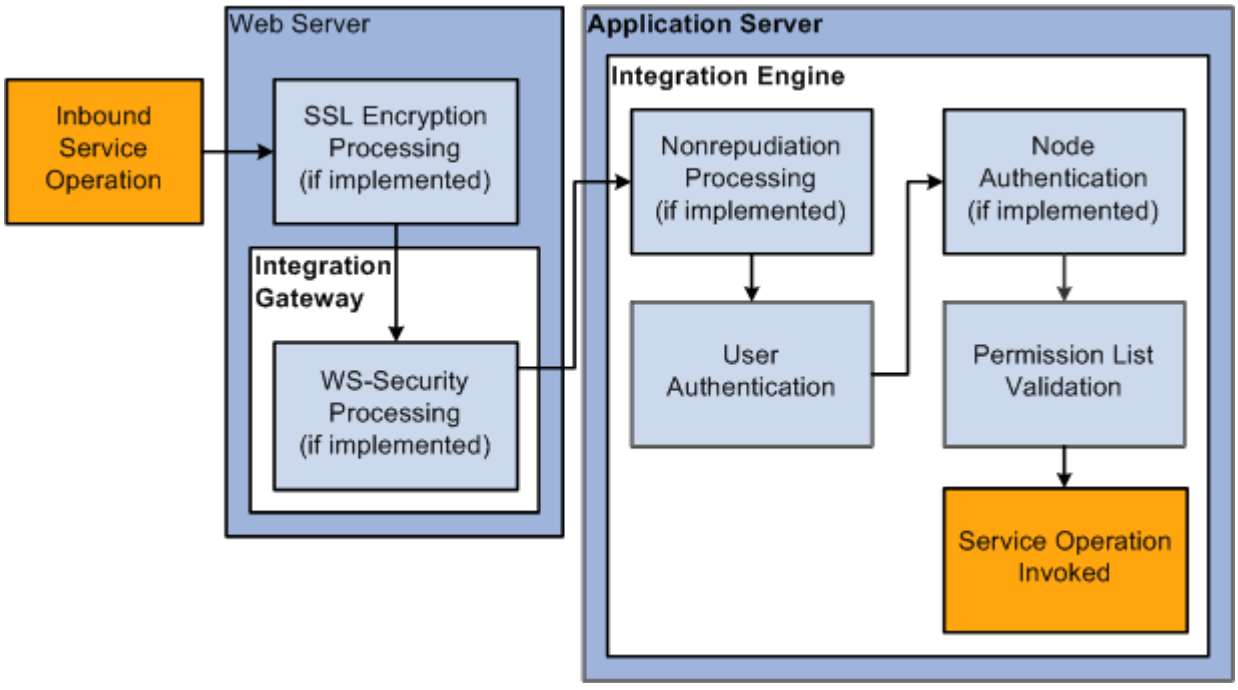
PeopleSoft Integration Broker applies the following security elements to outbound integrations:

Note. The elements are discussed in the order in which the system applies them.

User authentication	If the outbound service operation originates from a PeopleSoft (PIA) node, the user authentication process attaches the PeopleSoft authentication token to the service operation. If the service operation originates from an external (External) node, the model determines the user ID for the service operation and passes the information to the WS-Security framework so it can generate the UsernameToken for the outbound transaction.
Nonrepudiation	Nonrepudiation processing is performed.
Client authentication	Client authentication secures the connection between the PeopleSoft application server and the integration gateway on outbound transactions. You use digital certificates to secure this connection.
WS-Security	Outbound WS-Security processing includes generating the UsernameToken for the WS-Security SOAP header. This process may also involve encrypting and digitally signing the data, if specified in the WS-Security parameters on the node.
SSL encryption	SSL encryption on outbound integrations establishes a secure web server connection with an integration partner.

Inbound Integration Broker Security Processing

The following diagram illustrates security processing for inbound integrations to PeopleSoft Integration Broker:



Inbound PeopleSoft Integration Broker Security Processing

PeopleSoft Integration Broker applies the following security elements to inbound integrations:

Note. The elements are discussed in the order in which the system applies them.

SSL encryption	If the inbound service operation is encrypted, the integration gateway decrypts the data.
WS-Security	On inbound transactions, WS-Security processing includes validating a digital signature (if required), decrypting user information (if required), and passing the extracted user information to the integration engine for authentication.
Nonrepudiation	Nonrepudiation processing is performed.
User authentication	The system determines and validates the user ID associated with the inbound service operation.
Node authentication	If a node password is employed, the system validates that the inbound service operation contains the node password. If certificate authentication is employed, the system authenticates the node certificate.
Permission list validation	The system matches the user ID passed in with the service operation to the appropriate permission list.

Understanding Digital Certificates

This section provides an overview of :

- Digital certificates.
- Digital certificate authorities.
- Digital certificate installation elements.

Digital Certificates

A digital certificate is a form of electronic ID card that supports public key encryption technology. Each messaging participant generates a matched pair of encryption keys—a private key, which is never revealed or transmitted, and a public key, which is freely available to other participants. These keys are stored in a local file or repository called a *keystore*, and the public key is stored as part of a digital certificate. The certificate can be attached to a service operation to verify the sender's identity and to provide the recipient with the means to encode a response.

The following table lists the security technologies that require digital certificates and the digital certificate installation location for each of them. The table also lists the section in this chapter that discusses installing digital certificates for each of the technologies:

Security Technology	*Digital Certificate Installation Location*	*Section Describing How to Install Digital Certificates*	*Comments*
SSL encryption	Web server.	Setting Up Web Server SSL Encryption	Secures web server-to-web server connections.
WS-Security	Integration gateway.	Installing Integration Gateway-Based Digital Certificates	Secures web server-to-web server connections.
Client authentication	Integration gateway.	Installing Integration Gateway-Based Digital Certificates	Secures application server-to-integration connections.
Nonrepudiation	Application server.	Installing Application Server-Based Digital Certificates	Authenticates sender and receiver.
Certificated-based node authentication	Application server.	Installing Application Server-Based Digital Certificates	Authenticates sender.

Digital Certificate Authorities

A certificate authority (CA) is a trusted third-party organization or company that issues digital certificates used to create digital signatures and encryption keys. The role of the CA in this process is to guarantee the identity of the party granted the certificate. Usually, this means that the CA has an arrangement with a financial institution that provides information to validate the grantee's identity.

To install digital certificates for secure messaging, you must select a CA from whom to obtain the certificates. There are many CAs to choose from, and most of them do business on the World Wide Web. Some of the best known are:

- Verisign, Inc.
- Entrust Technologies.
- Baltimore Technologies.
- Thawte.

There are also numerous lesser known CAs, which might be appropriate if they are well known in a particular geographical region or industry. One of the systems participating in a secure integration might even serve as CA for the other participants. Each CA provides a unique set of security services and has its own way of handling digital certificates.

Before you implement secure messaging with PeopleSoft Integration Broker, investigate the available CAs, select one or more from whom you will obtain digital certificates, and familiarize yourself with their policies and procedures.

Digital Certificate Installation Elements

Whether you implement digital signature authentication, nonrepudiation, or SSL encryption, you need to use digital certificates. Although these security features require you to use a variety of programs and procedures, some characteristics of digital certificates—including the process of obtaining, installing, and configuring them—are common to all three features.

Depending on the security feature, you might install digital certificates in the keystore of an application server, a web server, or an integration gateway. An implementation of digital certificates on each of these entities involves the following elements:

- The entity's private and public encryption keys.
- A distinguished name (DN) for the entity.
- A certificate signing request (CSR).
- A certificate containing the entity's public encryption key, signed by a trusted CA.
- A root certificate from the trusted CA.

The following sections discuss these elements in more detail.

Public and Private Encryption Keys

For a given keystore, you generate private and public encryption keys simultaneously as a matching pair with software provided by the entity.

DN for the Entity

A DN is a property commonly used in security environments to uniquely identify a person, system, or network node. The DN is usually stored as a string of name-value attribute pairs separated by commas and spaces. You must provide the DN attribute values to generate a private key. These attributes include:

Common name (CN)	The name of the entity, expressed as a machine name, domain name, node name, or a name that you create, depending on the environment; for example, *QE_LOCAL*.
Organization unit (OU)	The part of the organization to which the entity belongs; for example, *Accounts Receivable.*
Organization (O)	The name of the organization or company; for example, *PeopleSoft.*
Locality (L)	The city or equivalent locality of the organization; for example, *Pleasanton.*
State (ST)	The state, province, or equivalent region of the locality; for example, *California.*
Country (C)	The country of the locality; for example, *US.*

CSR

A certificate signing request, or CSR, is a document that contains the entity's public key. The CSR is typically generated in Privacy Enhanced Mail (PEM) format, which is base64–encoded binary data. PEM is a standard text-based format for storing and transmitting digital certificates. You use the same software to generate the CSR that you use to generate the private-public key pair. The following example shows a CSR:

```
-----BEGIN NEW CERTIFICATE REQUEST-----
MIIBkTCB+wIBADBSMQswCQYDVQQGEwJ1czELMAkGA1UECBMCY2ExDTALBgNVBAcTBGhlcmUxCzAJ
BgNVBAoTAndlMQ0wCwYDVQQLEwR1bml0MQswCQYDVQQDEwJtZTCBnzANBgkqhkiG9w0BAQEFAAOB
jQAwgYkCgYEApaGAHNBjuByh8qXFCz33TgLzUjRm8S6tijit7fw23rKWyipQ0VgqeAD6eHr0pini
lyJPPOiJJ5fY0h2h78hOr8o+nJosTcqZL3jP+rSVick7qPPyXjcxP1UCGz/8RNykFDnbwjziwi+p
MesoWa8hfBss0ga2zZsmlV8Q4SyYE3UCAwEAAaAAMA0GCSqGSIb3DQEBBAUAA4GBACt0owTCngrU
/HAMAZgT/2O6hiZaD4OVBrgLYzmRvUiVhKOyTUzUv57ks7U6DQYt+rnWwNJtVbeAqO5eZiT7hXbj
Pwl8lGj+Adb6FGYOt4OhicZ0gNMHtURVop6iNJ9scxOmVcpkO0yX5f1rWFdZ0KZrWZSFGI6Lwdud
Hvbyvbpz
-----END NEW CERTIFICATE REQUEST-----
```

Signed Public Encryption Key From CA

The process of obtaining a signed public key certificate from a CA depends on the CA that you select. Typically, it requires you to paste the content of the PEM-formatted CSR into a form that you submit online. The CA then creates, digitally signs and returns a public key certificate to you. The CA will either email you the certificate or require you to download it from a specified web page. The certificate can be either PEM or the binary Distinguished Encoding Rules (DER) format. Following is an example of a PEM-formatted certificate:

```
-----BEGIN CERTIFICATE-----
MIICIDCCAcqgAwIBAgIQrDVQJKAAKLQR0/bIDJMSVDANBgkqhkiG9w0BAQQFADBy
MQswCQYDVQQGEwJVUzELMAkGA1UECBMCQ0ExEzARBgNVBAcTClBsZWFzYW50b24x
FzAVBgNVBAoTDlBlb3BsZVNvZnQgSW5jMRMwEQYDVQQLEwpQZW9wbGVUb29sMRMw
EQYDVQQDEwpQZW9wbGVUb29sMB4XDTAwMDMxMDIxMTIzNVoXDTA1MDMxMDIxMTIz
NVowcjELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAkNBMRMwEQYDVQQHEwpQbGVhc2Fu
dG9uMRcwFQYDVQQKEw5QZW9wbGVTb2Z0IEluYzETMBEGA1UECxMKUGVvcGxlVG9v
bDETMBEGA1UEAxMKUGVvcGxlVG9vbDBcMA0GCSqGSIb3DQEBAQUAA0sAMEgCQQCy
o44wplb57M272GRP3sC4TtLm/MD1G9osRjG9BWnsjjTij9GNi6Rnf9cNxkj+AGQY
gnE3P7lp9rYN6GQxPldNAgMBAAGjPDA6MAsGA1UdDwQEAwIBxDAMBgNVHRMEBTAD
AQH/MB0GA1UdDgQWBBSkFZJ1Dtt5uE6muLRN3rwRPsUCsTANBgkqhkiG9w0BAQQF
AANBAJec3hFPS2SLSDtfLI9mSA7UL1Vgbxr5zZ4Sj9y4I2rncrTWcBqj7EBp9n/Z
U/EwDEljVbE8SSDYr1Emgoxsr4Y=
-----END CERTIFICATE-----
```

Root Certificate

The root certificate contains the CA's digitally signed public key. It's also known as a *chain file* or a *signer certificate.* The process of obtaining a root certificate from a CA depends on the CA. The CA typically sends an email with the certificate or requires you to download it from a specified page.

The signed public key certificate also contains an embedded copy of the CA's root certificate, which you can export.

Installing Application Server-Based Digital Certificates

This section discusses how to:

- Install application server-based digital certificates.
- Access certificate properties.
- Export and convert certificates.

Understanding Installing Application Server-Based Digital Certificates

This section discusses how to install digital application server-based digital certificates.

Use the procedures discussed in this section for generating and installing digital certificates for use with nonrepudiation and certificated-based node authentication. Installing digital certificates for these security technologies requires that you install digital certificates in the application server keystore on each system participating in an integration.

However, while the process for generating application server-based digital certificates is the same for nonrepudiation and certificate-based node authentication technologies, generate and install separate certificates for each technology.

To install application server-based digital certificates on the PeopleSoft system use the Digital Certificates page (ADMINISTER_CERTS). This page enables you to:

- Install root certificates.
- Install signed public key certificates.
- Install a remote certificate.
- Export a certificate.

To obtain and import a local node certificate, use the Request New Certificate page (CERT_REQ_SBP).

Certificate Types

Each node requires three types of certificates:

- One root certificate from a trusted CA.

  This certificate contains the CA's digitally signed public key. Each root certificate is stored in a record of type *Root CA* in the keystore.

- One certificate containing the default local node's public key, signed by the same trusted CA.

  The CA's root certificate must be installed before you install the default local node's certificate, which is stored in a record of type *Local Node* in the keystore.

- One or more certificates containing the public keys of the remote nodes that participate in nonrepudiation or certificate-based node authenticated messaging.

  Each of these certificates is stored in a record of type *Remote.*

Remote Node Certificates

Any participating third-party system must have a set of certificates complementary to those installed at the PeopleSoft nodes.

Installing Application Server-Based Digital Certificates

This section discusses how to:

- Add CA authorities and install root certificates.
- Install signed public key certificates.
- Resolve root certificate mismatches.
- Install remote certificates.

Adding CA Authorities and Installing Root Certificates

PeopleSoft delivers a number of root certificates. Before you begin this process, check to see if your root certificate already exists. If it does, there is no need to perform this step.

If your root certificate does not exist, contact your CA for information on how to obtain the root certificate for importing into PeopleSoft.

To install a new root CA certificate:

1. Select PeopleTools, Security, Security Objects, Digital Certificates. The Digital Certificates page displays.
2. Add a CA authority:
   a. Click the plus button (+). A new row appears.
   b. From the Type drop-down list, select Root CA.
   c. In the Alias field, enter the alias name for the certificate.
   d. In the Issuer Alias field, enter an alias for the issuer. Click the Lookup button to select the certificate alias as the issuer alias.
3. Add the root certificate.
4. Click the Add Root link near the plus button (+). The Add Root Certificate page displays.
5. Copy the contents of the certificate into the text box.

   You must include the begin section (-----BEGIN CERTIFICATE-----) and end section (-----END CERTIFICATE -----).
6. Click the OK button.
7. Click the Refresh button.

Install Signed Public Key Certificates for Application Server-Based Digital Certificates

To section discusses how to:

- Add local node certificates to the PeopleSoft system and generate CSRs.

- Submit local node certificates to CAs for signing.
- Import signed local node certificates into the PeopleSoft system.

To install a signed public key certificate, you must define a local node certificate row in the keystore, then obtain the signed certificate from a CA whose root certificate is installed. To do this, you generate a CSR, submit the CSR to the CA, then retrieve and import the content of the signed certificate into your certificate row.

To add a local node certificate and generate a CSR:

1. Select PeopleTools, Security, Security Objects, Digital Certificates. The Digital Certificates page displays.
2. Click the plus button (+). A new row appears.
   a. From the Type drop-down list, select Local Node.
   b. In the Alias field, enter the name of the local node.

      Note. The name you enter must exactly match the name of the local node.

   c. In the Issuer Alias field, click the Lookup button to select the issuer alias.
3. At the end of the row, click the Request link. The Request New Certificate page displays.
4. In the Subject Information section, enter the following information:

   These fields represent attributes of the default local node's DN. The CA to whom you submit the CSR might require values for any or all of the fields. The DN is also stored on the Detail page of the local node certificate. For the common name, enter the name of the PeopleSoft Integration Broker default local node.

Company Name.	Enter the default local node name (with no underscore).
Org Unit(organizational unit)	Enter the name of the organizational unit.
Organization	Enter the name of the organization.
Locality	Enter the location of the organization.
State/Province	Enter the state or province name.
Country	Enter the two-character country code.

5. In the Key Pair Information section, enter the following information:
   a. From the Algorithm drop-down list, select MD5 with RSA Encryption.
   b. From the Key Size drop-down list, select 1024.
6. Click the OK button.

   In addition to generating the CSR, which contains the default local node's public key, this step also creates the matching private key, which is automatically installed in the same row of the node's keystore.

To submit a local node certificate for signing:

1. After you click the OK button as described in the previous section, the CSR is generated. Copy the CSR and submit it to your CA for signing.

   The process of obtaining digital certificates varies, depending on the CA. Typically, a CA requires you to paste the content of the PEM-formatted CSR into a form that you submit online.

   The CA may send you the signed public key certificate by email or require you to download it from a specified web page.

   When you submit the CSR for signing, you must include the begin section (-----BEGIN NEW CERTIFICATE REQUEST-----) and the end section (-----END NEW CERTIFICATE REQUEST-----).

2. When you receive the signed certificate back, copy it to a temporary directory. For example:

   ```
   c:\temp\newcert.cer
   ```

After you generate a CSR for the local node certificate and obtain a signature, you import the signed certificate into PeopleSoft.

To import signed local node certificates into a PeopleSoft system:

1. Select PeopleTools, Security, Security Objects, Digital Certificates. The Digital Certificates page displays.
2. Locate the row that contains the local certificate.
3. At the end of the row, click the Import link. The Import Certificate page displays.
4. Open the signed certificate you received back from the CA, copy it and paste it into the text box. The content you paste must include the begin section (-----BEGIN CERTIFICATE-----) and end section (-----END CERTIFICATE-----).
5. Click the OK button.
6. Click the Refresh button.

Three outcomes are possible:

- The Digital Certificates page appears and the new certificate's row now contains a Detail link. In this case, the certificate has been successfully installed, and you can proceed to install remote certificates for the node.

  Note. The new certificate's row may contain a different issuer alias than the one that you selected for it. This indicates that the keystore contains a root certificate signed by the same CA that signed the new certificate, but it wasn't the one with the issuer alias that you selected (the issuer alias of a root certificate doesn't always reflect which CA actually signed the certificate). PeopleSoft Integration Broker has changed the issuer alias for the new certificate to correctly reflect which root certificate is its parent.

- The following message may appear: *Could not decode PEM-formatted certificate data.* This indicates either that the pasted content isn't formatted properly as a certificate, or that the certificate is not yet valid.

  Every signed digital certificate has a period of time during which it can be used, specified by its internal timestamp fields, Valid From and Valid To, which are set by the signing CA. The timestamps were inserted by the CA's certificate server. You can't import the certificate content until the Valid From time has passed on your default local node's application server, which may lag by several minutes, depending on the relative clock accuracy of the two servers. Note that time zones are automatically accounted for and have no effect on this issue. You must examine theValid From field in the certificate's properties dialog box to determine when the certificate can be imported.

  See Chapter 11, "Setting Up Secure Integration Environments," Accessing Certificate Properties, page 170 .

- The following message may appear: *The certificate signature is not valid. The certificate is corrupt or has been modified.* This indicates either that the certificate has been tampered with, or that the keystore contains no root certificate signed by the same CA.

  The issuer alias of a root certificate doesn't always reflect which CA actually signed the certificate. Therefore it's possible that the CA to which you submitted your CSR didn't sign any of your installed root certificates. The local certificate in your keystore must be accompanied by a root CA certificate signed by the same CA.

Resolving Root Certificate Mismatches

To import a signed public key certificate to the application server keystore as a row of type *Local Node* on the Digital Certificates page, a root certificate signed by the same CA that signed the public key certificate must already exist as a *Root CA* row on that page.

If you cannot import a signed public key certificate because no matching root certificate exists, you can resolve the deficiency by installing the root certificate of the CA that *did* sign your public key certificate. Then you obtain a new signed public key certificate from that CA.

To resolve a root certificate mismatch:

1. Export the embedded root certificate from the signed public key certificate file.

   See Chapter 11, "Setting Up Secure Integration Environments," Exporting and Converting Certificates, page 170.

2. Define a new root CA certificate in the keystore.

   Refer to the previous procedure for establishing a root certificate.

3. Delete the local node row from the keystore's Digital Certificates page.

4. Add a new local node certificate to the keystore using the same issuer alias as the new root CA certificate.

   Refer to the previous steps for installing a signed public key certificate.

Installing Remote Certificates for Application Server-Based Digital Certificates

To section discusses setting up remote certificates for nonrepudiation and certificated-based node authentication and describes how to:

- Export remote node certificates.

- Add remote node CAs and import remote node certificates into the local node system.

To establish two-way authentication or nonrepudiation, each node must possess copies of the other participating nodes' public keys. You accomplish this with a certificate row of type *Remote* in the default local node's application server keystore, which contains a certificate exported from the row defined as *Local Node* in a remote node's keystore. You define one remote certificate for each participating remote node.

Note. Each remote certificate is a copy of the local node certificate and is installed on the remote node that it represents. As a result, you must first establish a root CA certificate and install a local node certificate on node A before you can export a copy of that certificate to node B. The simplest approach is to first install a certificate of type *Root CA* and a certificate of type *Local Node* on each of the participating nodes. Then you can export each of the local node certificates and import them to the other nodes as type *Remote*.

The following requirements apply:

- The remote system's local node certificate must already be installed.

  Refer to the previous steps for installing a signed public key certificate.

- The local system must have a root certificate installed with the same issuer alias (and actual issuer) as the remote system's local node certificate.

  Refer to the previous steps for establishing a root certificate.

Note. For the purposes of this discussion, assume that both local and remote nodes are PeopleSoft applications. If the remote node is a third-party system, the same requirements must still be satisfied—the third-party system must provide a copy of its signed public key certificate to the PeopleSoft node.

To export a remote node certificate:

1. On the remote node system, select PeopleTools, Security, Security Objects, Certificates. The Digital Certificates page displays.
2. Locate the row that contains the default local node, and click the Detail link at the end of the row. The Certificate Details page displays.
3. Click the Export button and copy the content in the edit box.
4. Click Cancel.

To add a remote node CA and import a remote node certificate into the local node system:

1. On the local node system, select PeopleTools, Security, Security Objects, Certificates. The Digital Certificates page displays.
2. Click the plus button (+). A new row appears.
   a. From the Type drop-down list, select Remote Node.
   b. In the Alias field, enter the name of the remote node.

      Note. The name you enter must exactly match the name of the remote node.

   c. In the Issuer Alias field, click the Lookup button to select the issuer alias.

3. Click the Refresh button.
4. At the end of the remote node row, click the Import link. The Import Certificate page displays.
5. Paste the certificate that you exported in the previous section into the text box. You must include the begin section (-----BEGIN CERTIFICATE-----) and the end section (-----END CERTIFICATE-----).
6. Click the OK button.
7. Click the Refresh button.

Accessing Certificate Properties

When you need to install a signed public key certificate in a keystore, you need the issuing CA's root certificate in the keystore as well. Your public key certificate is more than a single certificate; the same file contains the issuing CA's root certificate as well. If you do not receive a separate root certificate from the CA, you can access it from the public key certificate properties.

When you need to export a root certificate or examine the certificate's valid dates—or when you need to convert a certificate between DER and PEM formats—use the security extensions on a Windows machine to access the certificate properties dialog box .

To access certificate properties:

1. Double-click any certificate file with a .DER (binary format) extension or a .CER (PEM format) extension.

   This invokes the Windows extensions for security management, which open a dialog box so you can inspect the certificate properties.

2. (Optional.) Access the properties of the embedded root certificate.

   a. Select Certification Path.

      A tree structure appears, showing the hierarchical chain of trust between the public key certificate and its issuer root certificate. Your certificate has the common name that you supplied for it, and the issuer root certificate (its parent) has the name of its issuing CA.

   b. Select the root certificate, and click View Certificate.

      A dialog box display the properties of the root certificate.

3. (Optional.) Select Details.

   A list of fields appears. Click a field name to examine its value. This is especially useful for determining the certificate's Valid From and Valid To date and time.

Exporting and Converting Certificates

You might need to export an embedded root certificate or convert an existing certificate from DER format to PEM format. You can export certificates from:

- DER or PEM formatted certificate files.

- Certificate rows in a PeopleSoft application server keystore.

To export or convert a certificate from a file:

1. Access the properties dialog box of the certificate to export or convert.

   See Chapter 11, "Setting Up Secure Integration Environments," Accessing Certificate Properties, page 170.

2. In the certificate properties, select Details, then click Copy to File.

   The Certificate Export Wizard launches.

3. Click Next, then select a format.

   Base64-encoded X.509 (.CER) is the PEM format option, which is recommended. The *DER encoded binary X.509 (.CER)* option may also work, depending on the environment.

4. Click Next, and then browse to select a location and file name for the new certificate file.

   Specify the same location as the certificate. Ideally, you should give an exported root certificate file the same name as the issuing CA.

5. Click Next, then Finish to save the root certificate file.

   A message indicates when the export is successful.

To export a certificate from an application server keystore:

1. In the PeopleSoft Pure Internet Architecture, sign on to the application database and select PeopleTools, Security, Security Objects, Digital Certificates.

   The Digital Certificates page appears.

2. Click the Detail link of the desired certificate, then click Export.

   The Export Certificate page appears, containing the exportable certificate content in a long edit box.

3. Copy the entire certificate content and sign out of the database.

   Note. Save this certificate content to a file with a .CER extension.

Installing Integration Gateway-Based Digital Certificates

This section provides an overview of integration gateway-based digital certificates and discusses how to:

- Generate private and public key pairs.
- Generate CSRs.
- Obtain signed root certificates.
- Import signed root certificates.
- Specifying the keystore location for WS-Security.

- Encrypting keystore passwords for WS-Security.

Understanding Integration Gateway-Based Digital Certificates

Use the procedures discussed in this section for generating and installing digital certificates for use with client authentication and WS-Security.

Installing digital certificates for these security technologies requires that you install digital certificates in integration gateway keystores. However, the keystore locations where you install these certificates is different for each technology.

Also note that while the process for generating integration gateway-based digital certificates is the same for client authentication and WS-Security technologies, generate and install separate certificates for each technology.

Elements of Integration Gateway-Based Digital Certificates

To set up integration gateway-based digital certificates, you use a Java-based Keytool command line utility provided with PeopleSoft Integration Broker to install digital certificates in the integration gateway keystore.

The integration gateway requires the following elements:

- The gateway's private key.
- A certificate containing the gateway's public key, digitally signed by a trusted CA.
- A root certificate from the CA that signed the gateway's public key.

With Keytool, you generate a private-public key pair, which is automatically inserted in a gateway keystore that in created with the PeopleSoft Pure Internet Architecture installation in the web server directory structure.

The location of Keytool is <PIA_HOME>\webserv\<DOMAIN>\keystore\.

You generate a PEM-formatted CSR that contains the gateway's public key. You submit the CSR to the selected CA. The CA creates, digitally signs, and returns your gateway's public key certificate to you. This certificate also contains a signed copy of the CA's root certificate. These certificates may be in standard DER-encoded binary format, or they can be converted to PEM format if necessary.

You then install both signed certificates in the gateway keystore. In addition, you register them and the private key with the web server so that it can recognize and use them.

Keytool Utility

You may have previously installed software on the gateway server machine that included a distribution of the Keytool utility. To install digital certificates for client authentication SSL and WS-Security, be sure to use a copy of Keytool that was provided as part of the Java Runtime Environment (JRE). Use the copy of Keytool that was installed with either the PeopleTools application server or the web server. You can find Keytool in <PS_HOME>\jre\bin. You can also find it in the web server directory structure by searching for Keytool.exe (Windows) or keytool.sh (UNIX).

The basic syntax of Keytool is as follows:

```
keytool -command
```


Each command can be followed by a variety of options. Both the command and the keyword for each option that you invoke with it must be preceded by a hyphen, and most options must be followed by a value. If you enter *keytool* and hit Enter, , a list of all commands and their options is displayed. Keytool provides more than a dozen commands, but you'll use only a few for this task:

```
keytool -genkey
keytool -certreq
keytool -import
```

This section outlines only the basic steps required to install the certificates and keys that you need. You can obtain complete documentation for Keytool from Sun Microsystems.

See http://java.sun.com.

wss.properties File

The wss.properties file stores keystore location information and password information for WS-Security digital certificates.

When installing digital certificates for WS-Security, you must specify the location of the keystore in this file.

You can also store an encrypted copy of the keystore password in this file.

The location of the file is <PIA_HOME>\webserv\<DOMAIN>\peoplesoft\applications\PSIGW.war\WEB-INF\classes.

Generating Private and Public Key Pairs

To generate a key pair:

1. Open a command prompt and navigate to the location of the gateway keystore file.

   The location is <PIA_HOME>\webserv\<DOMAIN>\keystore.

2. Issue the following command (substituting the appropriate path for Keytool, if necessary):

```
PeopleTools_home\jre\bin\keytool -genkey -alias key_alias -keyalg RSA
-keysize keysize -dname "CN=cName, OU=orgUnit, O=org, L=locality,
ST=state, C=country" -keypass key_password -keystore pskey
-storepass password
```

Provide values for the options as follows:

alias Specify the name of the local default node. The private key associated with this alias is used for generating digital signatures.

Note. The value you enter must *exactly* match the name of the local default node.

key_alias Specify a name for the key pair. For example:

```
My_GW_Client_Key
```

You also enter this value in the integrationGateway.properties file.

keysize Specify one of the following values for the key size:

- *1024*: This specifies a 1024-bit key, which provides 128-bit encryption. This is the default value.
- *512*: This specifies a 512-bit key, which provides 40-bit encryption.

dname "CN=cName, OU=orgUnit, O=org, L=locality, ST=state, C=country" Specify the gateway's DN attributes. The DN attributes are name-value pairs separated by commas and spaces, and they are enclosed in quotes as a single string. If a value includes a comma, you must precede the comma with a backslash escape character; for example:

```
O=PeopleSoft\, Inc.,
```

You must supply the DN attributes in the order shown. Although their values can be arbitrary, you should supply the appropriate real-world information.

key_password Enter a password of your choice for the key pair. It must be at least six characters long. You also enter this value in the integrationGateway.properties file.

The key pair is generated and must be imported into the keystore.

Generating CSRs

While you are at the command line in the gateway keystore directory, issue the following command:

```
PeopleTools_home\jre\bin\keytool -certreq -alias key_alias
-file csr_filename -keypass key_password -keystore pskey
-storepass password
```

Provide values for the options as follows:

alias Specify the name of the local default node. The private key associated with this alias is used for generating digital signatures.

Note. The value you enter must *exactly* match the name of the local default node.

key_alias Enter the name of the key pair that you created previously; for example:

```
My_GW_Client_Key
```

csr_filename Specify the name of the file that contain the CSR; for example:

```
My_GW_Client_Key.csr
```

You can also include a path for the file to create it in a different location than the keystore.

key_password Enter the password that you specified when you created the key pair.

The CSR file appears in the location and with the name that you specified.

Obtaining Signed Root Certificates

You need to obtain a signed certificate from the selected CA. The signed certificate contains your gateway's public key. The process of obtaining digital certificates varies, depending on the certificate authority that you select. Typically, a CA requires you to paste the content of the PEM-formatted CSR into a form that you submit online. The CA may send you the signed public key certificate by email, or it may require you to download the certificate from a specified page. The CA may also provide its root certificate or instructions for retrieving it.

Use the appropriate method for submitting a CSR for signing as determined by your CA.

When you do submit the CSR for signing the content you provide must include the begin section (-----BEGIN NEW CERTIFICATE REQUEST-----) and end section (-----END NEW CERTIFICATE REQUEST-----) of the CSR.

The CA will return the signed certificate to you.

Save the certificates to the location of the keystore file.

The location is <PIA_HOME>\webserv\<DOMAIN>\keystore.

Importing Signed Root Certificates

The public key certificate includes more than a single client certificate; the same file contains the issuing CA's root certificate as well. If you do not receive a separate root certificate from the CA, you must export it from the public key certificate.

To import signed root certificates:

1. Open a command prompt and navigate to the gateway keystore file.

   The location is <PIA_HOME>\webserv\<DOMAIN>\keystore.

2. Issue the following command (substituting the appropriate path for Keytool, if necessary):

```
<PS_HOME>\jre\bin\keytool -import -trustcacerts -alias root_cert_alias
-file root_cert_filename -keystore pskey -storepass password
```

This command imports the signed root certificate into the gateway keystore. Provide values for the options as follows:

root_cert_alias Specify the alias to use on your gateway to refer to the root certificate; for example:

```
"Root SGC Authority"
```

root_cert_filename Enter the name of the root certificate file that you received from the CA or exported from the public key certificate; for example:

```
"Root SGC Authority.cer"
```

3. While at the command line in the gateway keystore directory, issue the following command:

```
<PS_HOME>\jre\bin\keytool -import -alias key_alias
-file client_cert_filename -keypass key_password -keystore pskey
-storepass password
```

This command imports the signed public key certificate into the gateway keystore. Provide values for the options as follows:

alias Specify the name of the local default node. The private key associated with this alias is used for generating digital signatures.

Note. The value you enter must *exactly* match the name of the local default node.

---

key_alias Enter the name of the key pair that you created previously, for example:

```
My_GW_Client_Key
```

client_cert_filename Specify the name of the newly received public key certificate; for example:

```
My_GW_Client_Key.cer
```

key_password Enter the password that you specified when you created the key pair.

Specifying the Keystore Location for WS-Security

After you install digital certificates for WS-Security, you must specify the keystore location in the wss.properties file.

To specify the keystore location for WS-Security:

- Open the wss.properties file.

  The location of the file is <PIA_HOME>\webserv\<DOMAIN>\peoplesoft\applications\PSIGW.war\WEB-INF\classes.

- Set the following property equal to the location and file name of the keystore where you installed the integration gateway-based digital certificates.

  ```
  org.apache.ws.security.crypto.merlin.file
  ```

  For example:

  ```
  org.apache.ws.security.crypto.merlin.file=c:/<PIA_HOME>/<webserv>/
  <DOMAIN>/keystore/pskey
  ```

  Note. When entering the path to the keystore, use you must use either double-backslashes ("\\") or forward slashes ("/") as path separators. Do not use backslashes ("\") as path separators for directory names in the wss.properties file. Backslashes are misinterpreted as escape characters by the Java processes that access the file.

- Save the changes.

Encrypting Keystore Passwords for WS-Security

This section discusses how to encrypt the password for the keystore that contains digital certificates for WS-Security.

Understanding Encrypting Keystore Passwords for WS-Security

When working with the WS-Security digital certificates, PeopleSoft recommends that you encrypt the keystore password in the wss.properties file using the PSCipher utility.

Encrypting the WS-Security Keystore Password

To encrypt the WS-Security keystore password, making sure to write down the encrypted output.

1. Encrypt the WS-Security keystore password using the PSCipher utility.

   See Chapter 5, "Managing Integration Gateways," Encrypting Passwords Using the PSCipher Java Utility, page 45.

2. Access the wss.properties file.

   The location is <PIA_HOME>\webserv\<DOMAIN>\peoplesoft\applications\PSIGW.war\WEB-INF\classes.

3. Set the following property equal to the encrypted password you created using the PSCipher utility:

   ```
   org.apache.ws.security.crypto.merlin.keystore.password
   ```

   The following example shows an encrypted password entered for this property:

   ```
   org.apache.ws.security.crypto.merlin.keystore.password=UWZzB57U6SE=
   ```

4. Save the changes.

Installing Web Server-Based Digital Certificates

This section discusses how to:

- Install digital certificates on Oracle WebLogic web servers.
- Install digital certificates on IBM WebSphere web servers.

Understanding Installing Web Server-Based Digital Certificates

You must install web server-based digital certificates to implement web server SSL encryption.

You use utilities provided with the Oracle WebLogic or IBM WebSphere software to install web server-based certificates for SSL encryption. This authentication secures inbound messages. The web server requires three elements:

- The web server's private key.
- A certificate containing the web server's public key, digitally signed by a trusted certificate authority (CA).
- A root certificate from the CA that signed the web server's public key.

The information in section outlines the basic steps required to obtain and install the certificates and keys that you need. Oracle WebLogic and IBM WebSphere provide their own interface and methodology for establishing SSL encryption—you should refer to the documentation supplied with the web server software for detailed information about this process. In addition, refer to the information supplied by the selected CA.

Note. PeopleSoft delivers a number of certificate authorities and root certificates. If your certificate authority or root certificate is not listed, you need to add it to the PeopleSoft system.

You use the web server software to generate its own private key. At the same time, it also generates a certificate signing request (CSR), which contains the web server's public key. You submit the CSR to the selected CA, which creates, digitally signs, and returns your web server's public key certificate to you. This certificate might be in standard DER-encoded binary format; however, it can be converted to PEM format if necessary. You then install both signed certificates, and you register them and your private key with your web server, so that the web server recognizes and uses them.

Installing Digital Certificates for SSL Encryption on Oracle WebLogic

This section describes how to install digital certificates for SSL encryption for the Oracle WebLogic environment and discusses how to:

- Generate and import public keys.
- Generate private keys and CSRs.
- Submit CSRs to CAs for signing.
- Import signed private keys into keystores.

- Set up gateway private keys.
- Set up Oracle WebLogic Console for SSL.

Generating and Importing Public Keys (WebLogic)

Before you can generate and import public keys into PeopleSoft, you must access and download the signed public key from your CA. The process for accessing and downloading the signed public key varies, depending on your CA. Contact your CA for information on how to perform these tasks.

To generate and import public keys:

1. Place the public key from your CA in the keystore. The location of the keystore is:

   ```
   <PIA_HOME>\webserv\peoplesoft\keystore
   ```

2. Open PSKeyManager.

   a. Open a command prompt and navigate to <PIA_HOME>\webserv\<DOMAIN>.

   b. Enter the following at the prompt:

      ```
      pskeymanager -import
      ```

   c. At the Enter current keystore password prompt, enter the password and press Enter.

3. At the Specify an alias for this certificate prompt, enter the alias name and press Enter.

   The alias name you enter must be the same one you entered when you generated the private key.

4. At the Enter the name of the certificate file to import prompt, enter the path and name of the certificate to import, and press Enter.

5. At the Trust this certificate prompt, enter *Yes* and press Enter.

Generating Private Keys and CSRs (WebLogic)

You use PSKeyManager to generate private keys. PSKeyManager is a wrapper to Sun Microsystem's Keytool for managing keys and certificates.

While using PSKeyManager, press the Enter key to select any of the default values presented.

To generate the private key and the CSR on Oracle WebLogic:

1. Start PSKeyManager.

   a. Open a command prompt and navigate to <PIA_HOME>\webserv\<DOMAIN>.

   b. Enter the following at the prompt:

      ```
      pskeymanager -create
      ```

      PSKeyManager opens.

2. Enter the current keystore password and press Enter.

   The default password is *password.*

3. At the Specify an Alias for this Certificate <host_name>? prompt, enter the certificate alias and press Enter.

   The default certificate alias is the local machine name.

4. At the What is the common name for this certificate <host_name>? prompt, enter the host name for the certificate. For example:

   ```
   <host_name>.corp.peoplesoft.com
   ```

   Press Enter.

5. Enter the appropriate information at the following prompts. Press Enter after each entry.

   a. Organization unit.

   b. Organization.

   c. City of locality.

   d. State or province.

      You must spell out the entire state name. Do not enter an abbreviation.

   e. Country code.

   f. Number of days the certificate should be valid.

      The default value is *90.*

   g. Key size to use.

      The default value is *1024.*

   h. Key algorithm.

      The default value is *RSA.*

   i. Signing algorithm.

      The default value is *MD5withRSA.*

6. At the Enter a private key password prompt, enter the password or press Enter to use the keystore password.

7. Verify that the values you entered are correct, and press Enter. To go back and change any values, enter *No* and press Enter.

PSKeyManager generates a private key and provides the certificate signing request (CSR) that you will provide to the CA for signing. The following example shows a sample CSR.


```
-----BEGIN NEW CERTIFICATE REQUEST----- MIIBtDCCAR0CAQAwdDELMAk
GA1UEBhMCVVMxEDAOBgNVBAgTB0FyaXpvbmExEDAOBgNVBAcTB1Bob2VuaXgxFD
ASBgNVBAoTC1Blb3BsZVRvb2xzMRMwEQYDVQQLEwpQZW9wbGVzb2Z0MRYwFAYDV
QQDEw1NREFXU09OMDUxNTAzMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQC43
lCZWxrsyxven5QethAdsLIEEPhhhl7TjA0r8pxpO+ukD8LI7TlTntPOMU535qMGfk
/jYtG0QbvpwHDYePyNMtVou6wAs2yr1B+wJSp6Zm42m8PPihfMUXYLG9RiIqcmp2F
zdIUi4M07J8ob8rf0W+Ni1bGW2dmXZ0jGvBmNHQIDAQABoAAwDQYJKoZIhvcNAQEE
BQADgYEAKx/ugTt0soNVmiH0YcI8FyW8b81FWGIR0f1Cr2MeDiOQ2pty24dKKLUqI
hogTZdFAN0ed6Ktc82/5xBoH1gv7YeqyPBJvAxW6ekMsgOEzLq9OU3ESezZorYFdrQT
qsEXUp1A+cZdfo0eKwZTFmjNAsh1kis+HOLoQQwyjgaxYI=
-----END NEW CERTIFICATE REQUEST-----
```

The CSR is written in as a text file to the <PIA_HOME>\webserv\peoplesoft directory. The file name is *<host_name>_certreq.txt.*

Submitting CSRs to CAs for Signing (WebLogic)

After you generate the private key and a certificate signing request (CSR), you must submit the CSR to the certificate authority (CA) for signing.

The process of obtaining the signature varies, depending on the CA that you select. Typically, a CA requires you to paste the content of the PEM-formatted CSR into a form that you submit online. However, the CA may send the signed public key (root) certificate to you by email or require you to download it from a specified web page. The CA may also provide its root certificate or instructions for retrieving it.

Use the appropriate method to submit a CSR for signing as determined by your CA.

When you do submit the CSR for signing the content you provide must include the begin section (-----BEGIN NEW CERTIFICATE REQUEST-----) and the end section (-----END NEW CERTIFICATE REQUEST-----) of the CSR.

The CA will return the signed certificate to you that you must import into the keystore.

Importing Signed Private Keys into Keystores (WebLogic)

You use PSKeyManager to import a server-side private key into the keystore.

1. Open PSKeyManager.
   a. Open a command prompt and navigate to <PIA_HOME>\webserv\<domain>.
   b. Enter the following at the prompt:

      ```
      pskeymanager -import
      ```

   c. At the Enter current keystore password prompt, enter the password and press Enter.
2. At the Specify an alias for this certificate prompt, enter the alias name and press Enter.

   The alias name you enter must be the same one you entered when you generated the private key.
3. At the Enter the name of the certificate file to import prompt, enter the path and name of the certificate to import, and press Enter.
4. At the Trust this certificate prompt, enter *Yes* and press Enter.

Setting Up Gateway Private Keys (WebLogic)

To set up private keys for gateways, follow the procedures outlined in the following topics presented earlier in this section:

- Generating Private Keys and CSRs.
- Submitting CSRs to CAs for Signing.
- Importing Server-Side Private Keys into Keystores.

The only difference is that for the following prompts you enter names that are gateway-specific:

Prompt	Sample Values
Certificate alias.	Enter an alias, such as *PT850GATEWAY*.
Common name for this certificate.	Enter a name, such as *PT850GATEWAY*.

Setting Up Oracle WebLogic for SSL Encryption

To set up Oracle WebLogic for SSL:

1. Login to WebLogic Console.
    a. Open a web browser.
    b. In the URL or address field, enter *http://localhost/index.html* and press Enter. The Oracle WebLogic Server 8.1 Web Server Index Page displays.
    c. Click Access WebLogic Server Console. The signon page for WebLogic Server Administration Console appears.
    d. Enter the Username and Password and click *Sign In.* WebLogic Administration Console displays.

    The username and password are those that you specified when you installed PeopleSoft Pure Internet Architecture.
2. Navigate to the PIA server Configuration page.
    - In the WebLogic Server Console In the left navigation area, navigate to PeopleSoft, Servers, PIA. Or,
    - In the WebLogic Server Console, in the Domain Configuration section, click Servers. The Servers page displays. In the table that appears on the page, click the PIA link.
3. Click the Keystores and SSL tab.
4. In the Keystore Configuration section, on the right side of the page, click the Change link. The Specify Keystore Type page displays.
5. From the Keystores drop-down list, select *Custom Identity and Custom Trust.*
6. Click the Continue button. The Configure Keystore Properties page displays.

7. In the Custom Identity section complete the following fields:

   a. In the Custom Identity Key Store File Name field, enter *keystore/pskey*.

   b. In the Custom Identity Key Store Type field, enter *JKS*.

   c. In the Custom Identity Key Store Pass Phrase field, enter *password*.

   d. In the Confirm Custom Identity Key Store Pass Phrase field, enter *password* again.

   e. Click the Continue button. The Review SSL Private Key Settings page displays.

8. In the Review SSL Private Key Setting page, review the information and click the Continue button.

9. Click the Finish button. You will restart the web server at a later time. You are returned to the Keystore Configuration tab.

10. Scroll down the page to the Advanced Options section and click the Show link.

11. In the Server Attributes section, from the Two Way Client Cert Behavior drop-down list box, select *Client Certs Requested and Enforced*.

12. Click the Apply button.

13. Restart the web server.

Installing Digital Certificates for SSL Encryption on IBM WebSphere

This section describes how to install digital certificates for SSL encryption for the IBM WebSphere environment and discusses how to:

- Generate and import public keys.
- Generate private keys and CSRs.
- Submit CSRs to CAs for signing.
- Import signed private keys into keystores.
- Set up gateway private keys.
- Set up IBM WebSphere for web server SSL encryption.

Generating and Importing Public Keys (WebSphere)

Before you can generate and import public keys into PeopleSoft, you must access and download the signed public key from your CA. The process for accessing and downloading the signed public key varies, depending on your CA. Contact your CA for information on how to perform these tasks.

To generate and import a root certificate:

1. From the Key Database File menu, select Open PSKEY. The location is:

   ```
   <PIA_HOME>\webserv\<cell_name>_<node_name>_<server_name>\peoplesoft.ear\keystore⇒
   \pskey
   ```

2. Click the Download button and load the file to <PIA_HOME>\webserv\<DOMAIN>. For example:

   ```
   <PIA_HOME>\webserv\<DOMAIN>\<host_name>_PeopleTools.cer
   ```

3. In the Password field, enter *password.*
4. In the Key Database Content section, from the drop-down list select Signer Certificates.
5. Click the Add button to add a CA certificate.
6. Enter the following values:
   a. In the Data Type field, select or enter *Binary DER data.*
   b. In the Certificate File Name field, enter *<host_name>_PeopleTools.cer.*
   c. In the Location field, specify *<WAS_HOME>\ssl.*
7. Click the OK button and select a label.

Generating Private Keys and CSRs (WebSphere)

To generate private keys in IBM WebSphere you use IBM Key Management.

To generate server-side private keys and CSRs:

1. Open IBM Key Management.
   a. Open a command prompt and navigate to <WEBSPHERE_HOME>\appserver\bin.
   b. At the prompt, enter the following:

      ```
      Ikeyman
      ```

   c. Press the Enter key. IBM Key Management opens.
2. Select Key Database File, Open PSKEY.

   The location is:

   ```
   <PIA_HOME>\webserv\<cell_name>_<node_name>_<server_name>\peoplesoft.ear\
   keystore\pskey
   ```

3. Enter the password.
4. In the Key Database Content section, from the drop-down list select *Personal Certificate Requests.*
5. Click the New button. The Create New Key Certificate Request window opens.

6. Enter the appropriate information in the following required fields:

Key Label	Enter the host name.
Key Size	From the drop-down list select *1024.*
Common Name	Enter the host name for the certificate. For example: `<host_name>.corp.peoplesoft.com`
Organization	Enter the organization name.

7. In the Enter the name of a file in which to store the certificate request field, enter the location in Step 2.
8. Click the OK button. The window closes.

   In the Key Database Content section, the key label appears under the Personal Certificate Requests section.

IBM Key Management generates and writes the private key to <WAS_HOME>\ssl\certreq.arm.

Submitting CSRs to CAs for Signing (WebSphere)

After you generate the private key and a certificate signing request (CSR), you must submit the CSR to the certificate authority (CA) for signing.

The process of obtaining the signature varies, depending on the CA that you select. Typically, a CA requires you to paste the content of the PEM-formatted CSR into a form that you submit online. However, the CA may send the signed public key certificate to you by email or require you to download it from a specified web page. The CA may also provide its root certificate or instructions for retrieving it.

Use the appropriate method for submitting a CSR for signing as determined by your CA.

When you do submit the CSR for signing the content you provide must include the begin section (-----BEGIN NEW CERTIFICATE REQUEST-----) and end section (-----END NEW CERTIFICATE REQUEST-----) of the CSR.

The CA will return the signed certificate to you.

Importing Signed Public Keys into Keystores (WebSphere)

After you receive a signed certificate back from the CA, you must import it into the keystore.

To import server-side public keys into keystores:

1. Open IBM Key Management.
   a. Open a command prompt and navigate to <WEBSPHERE_HOME>\appserver\bin.
   b. At the prompt, enter the following:

      ```
      Ikeyman
      ```

   c. Press the Enter key. IBM Key Management opens.

2. In the Key Database Content section, from the drop-down list select *Personal Certificates.*
3. Click the Receive button. The Receive Certificate from a File box displays.
4. From the Data Type drop-down list, select *Base64-encoded ASCII Data.*
5. In the Certificate File Name field enter the name of the certificate to import or click the Browse button to locate the file.
6. In the Location field, enter the path to the certificate file.
7. Click the OK button.

   The Receive Certificate from a File box closes and the name of the certificate appears in the Personal Certificates section in IBM Key Management.

Setting Up Gateway Private Keys (WebSphere)

To set up private keys for gateways, follow the procedures outlined in the following topics presented earlier in this section:

- Generating Private Keys and CSRs.
- Submitting CSRs to CAs for Signing.
- Importing Server-Side Private Keys into Keystores.

The only difference is that for the following prompts you enter names that are gateway-specific:

Prompt	***Sample Values***
Certificate alias.	Enter an alias, such as *PT850GATEWAY.*
Common name for this certificate.	Enter a name, such as *PT850GATEWAY.*

Setting Up IBM WebSphere for Web Server SSL Encryption

Setting up IBM WebSphere for web server SSL encryption requires that you perform the following tasks:

- Configure SSL repertoires.
- Set up WebSphere servers for SSL encryption.
- Set up inbound Common Secure Interoperability (CSI) authentication.

To configure an SSL repertoire:

1. 1. Start the WebSphere Administration Console.

   The URL is http://localhost:9090/admin/.

2. 2. In the left navigation area, navigate to Security, SSL. The SSL Repertories page displays.
3. Click the New button. The SSL Configuration Repertoires page displays.

4. On the Configuration tab, enter values for the following fields:

    a. In the Alias field enter Web Container SSL.

    b. In the Key File Name field enter the location of the JKS file or the location of PSKey. For example:

    ```
    <PIA_HOME>\webserv\<cell_name>_<node_name>_<server_name>\peoplesoft.ear\
    keystore\pskey
    ```

    c. In the Key File Password field, enter the keystore password.

    d. In the Key File Format field, enter *JKS.*

    e. In the Trust File Name field, enter the location of the location of the JKS file or the location of PSKey.

    f. In the Trust File Password field, enter the certificate password.

    g. In the Trust File Format field, enter *JKS.*

    h. Clear the Client Authentication box, if selected.

    i. In the Security Level field, select High.

    j. Click OK.

5. Save the configuration.

To set up a WebSphere server for SSL encryption:

1. Open the WebSphere Administration Console, if it is not already open.

    The URL is http://localhost:9090/admin/.

2. In the left navigation area, select Servers, Application Servers and select the server with which you would like to work. The Application Servers page displays.

3. Click the name of the server that appears as a hyperlink on the page.

4. Click the Configuration tab.

5. In the Additional Properties section, click Web Container. The Web Container page displays.

6. In the Additional Properties section, click the HTTP Transports link.

7. Check the box of the row that contains the entry for the transfer you want to secure.

8. In the Hosts column click the asterisk (*). The HTTP Transports page displays.

9. In the Configuration panel in the General Properties section, for the SSL Enabled property check the Enable SSL box.

10. From the SSL drop-down list, select the desired SSL entry from the repertoire.

11. Click the OK button and save the changes.

To set up CSI authentication:

1. Open the WebSphere Administration Console, if it is not already open.

   The URL is http://localhost:9090/admin/.

2. In the left navigation area, navigate to Security, Authentication Protocol, CSIV2InboundAuthentication. The CSI Authentication ->Inbound page displays.

3. For Basic Authentication, select Supported.

4. For Client Certificate Authentication, select Required.

5. Save the changes and reboot the web server.

Implementing Web Server SSL Encryption

This section provides an overview of web server SSL encryption and discusses how to:

- Configure web server SSL encryption.
- Implement web server SSL encryption.

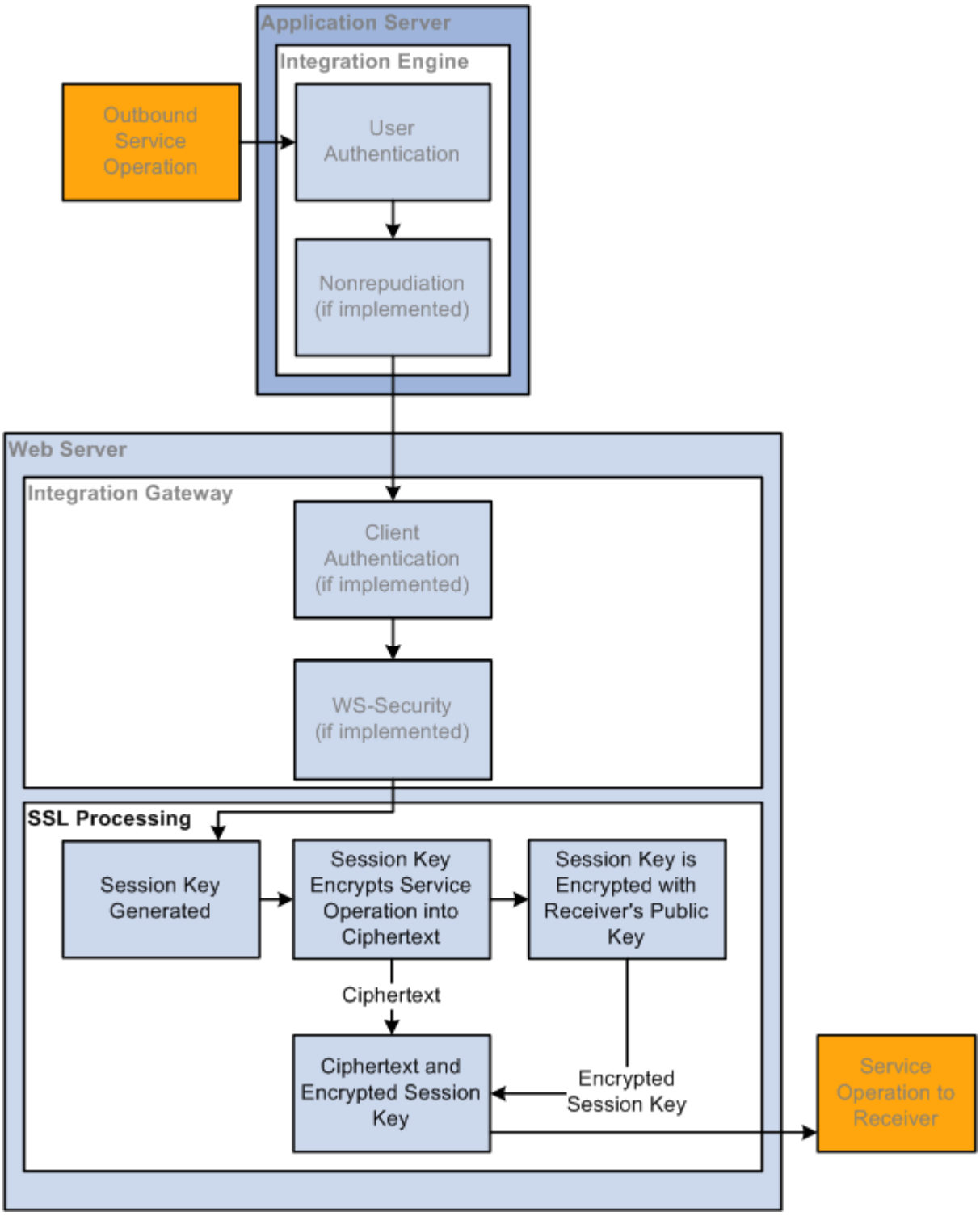
Understanding Web Server SSL Encryption

This section discusses:

- Outbound web server SSL encryption.
- Inbound web server SSL encryption.

Outbound Web Server SSL Encryption

The following diagram shows the processing that occurs on outbound transactions when web server SSL encryption is implemented:



Outbound Web Server SSL Encryption Processing

Before the integration starts, your integration partner generates a key pair that consists of a private key and a public key. The private key is placed in its web server keystore. The public key is placed in a digital certificate.

You contact the integration partner's site using a secured URL that begins with HTTPS. The integration partner's site responds by sending you its web server digital certificate, which contains the public key of the key pair it generated prior to initiating the integration.

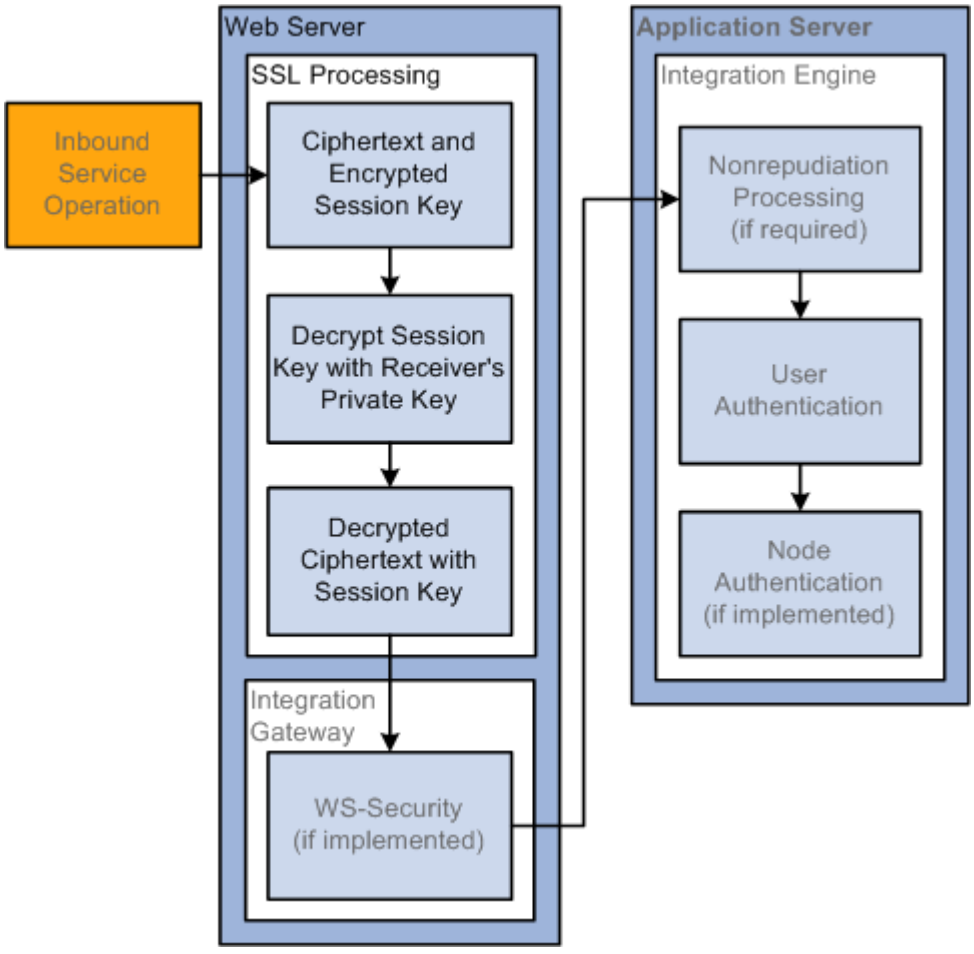
Your web server generates a session key to encrypt the plain text outbound request contents into ciphertext. Then the web server encrypts the ciphertext and session key using your integration partner's public key that was sent to you in the digital certificate.

The session is now secure and all communication is encrypted and can only be decrypted by you and your integration partner.

When the request arrives at your integration partner's web server, the integration partner's web server uses its private key to decrypt the ciphertext and session key. It then uses the session key to decrypt the ciphertext and extract the service operation contents in plain text.

Inbound Web Server SSL Encryption

The following diagram shows the processing that occurs on inbound transactions when web server SSL encryption is implemented.



Inbound Web Server SSL Encryption Processing

Before the integration starts, you generate a key pair that consists of a private key and a public key. You place the private key in your web server keystore and the public key gets placed in a digital certificate.

For inbound web server SSL encryption processing, your integration partner contacts you using a secured HTTPS URL. Your web server responds by sending the integration partner a web server digital certificate that contains your public key. The integration partner's web server goes through the outbound processing described in the previous section.

When the service operation arrives on your web server, it is one package that contains the ciphertext (encrypted service operation contents) and the encrypted session key that decrypts the ciphertext.

Your web server uses its private key to decrypt the ciphertext and session key. It then uses the session key to decrypt the ciphertext into a plain text service operation.

Prerequisites for Implementing Web Server SSL Encryption

You must set up web server-based digital certificates to implement web server SSL encryption.

Configuring Web Server SSL Encryption

Configuring web server SSL encryption involves the following tasks:

- Supply the digital certificates containing the public and private keys required for encrypted transactions. You install these certificates in the web server keystore. You configure the web server to recognize and use its installed certificates for SSL transactions.
- Edit the Integration Gateway Certificates section of the integrationGateway.properties file to convey parameters for the web server certificates that you installed in the gateway keystore.

Integration Gateway Properties File Parameter	***Description***
ig.certificateAlias	Certificate alias.
ig.certificatePassword	Certificate alias password.

See Chapter 5, "Managing Integration Gateways," Using the integrationGateway.properties File, page 41.

Implementing Web Server SSL Encryption

For outbound transactions you must change the value of the HTTP target connector's PRIMARYURL;URL property to start with *https://* instead of *http://*. You can apply this setting on a node-by-node basis, or apply it to the gateway as a whole, which will use it as the default setting for all nodes. The HTTP target connector makes the necessary SSL connection at runtime.

Receipt of HTTPS requests is nearly automatic. When the integration gateway's HTTP listening connector receives an HTTPS request, it is forwarded to the default local node for authentication.

Implementing Web Services Security

This section provides and overview of WS-Security and WS-Security processing in PeopleSoft Integration Broker. It also discusses prerequisites for implementing WS-Security in PeopleSoft Integration Broker and discusses how to:

- Implement WS-Security for inbound integration (Username Tokens).
- Implement WS-Security for inbound integration (SAML Tokens).
- Implement WS-Security for outbound integration (Username and SAML Tokens).
- Override node-level WS-Security settings on routing definitions.
- Implement WS-Security on services consumed using the Consume Web Services wizard.

This section also describes WS-Security configuration options for outbound integrations and provides examples for WS-Security SOAP message headers.

Understanding Implementing WS-Security in PeopleSoft Integration Broker

This section provides an overview of implementing WS-Security in PeopleSoft Integration Broker.

WS-Security Standard Supported

PeopleSoft implements WS-Security in accordance with Oasis standards.

Within this framework, PeopleSoft implements:

- Username tokens.
- SAML tokens.

The PeopleSoft implementation of WS-Security supports:

- Clear-text username token. (Password is optional.)
- Digitally signed username token.

  Digital signatures apply to the SOAP message header and SOAP message body.
- Encrypted username token.

  You specify to encrypt the SOAP header only, SOAP header and message body, or the message body only
- Encrypted SAML assertion token.

  You specify to encrypt the SOAP header only, SOAP header and message body, or the message body only

Please visit the MyOracle Support website for information about the current versions of the WS-Security standards, profiles, and namespaces supported by PeopleTools.

See http://www.oracle.com/support/index.html.

UsernameToken Profile

A UsernameToken is the means of identifying a requestor by user name to authenticate the user's identity to the web service provider. A password may also be used in conjunction with the user name.

The UsernameToken is supplied in the <UsernameToken> element in the WS-Security SOAP header that gets added to an inbound or outbound service operation when WS-Security is implemented. The elements included in the credential are discussed in the following section.

On outbound service operations, the values that the PeopleSoft system populates in the UsernameToken profile can be derived from an external user ID that you specify on the node definition for the external node. It can also be derived from the default user ID specified on the external node definition. In addition, you can choose to digitally sign and encrypt this information.

SAML Token Profile

The Security Assertion Markup Language (SAML) is an XML-based framework for exchanging security information. All SAML tokens include the following common information as defined by Oasis standards:

- Issuer ID.
- Subject.
- Name.
- Subject confirmation.
- Conditions under which the assertion is valid.

  Example of these conditions are *NotBefore* and *NotOnOrAfter*.

This security information is expressed in the form of assertions about subjects, where a subject is an entity that has an identity in some security domain.

The following pseudocode shows an example of a SAML token:

```
<Assertion AssertionID="d9aeaa4c1126df5ee0c6df64fdf961b1" IssueInstant="2008-05-
  14T18:18:47.246Z" Issuer=".peoplesoft.com" MajorVersion="1" MinorVersion="1"
  xmlns="urn:oasis:names:tc:SAML:1.0:assertion" xmlns:saml="urn:oasis:names:tc:
  SAML:1.0:assertion" xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol">
  <Conditions NotBefore="2008-05-14T18:18:47.184Z" NotOnOrAfter="2008-05-14T18:
   28:47.184Z"/>
   <AuthenticationStatement AuthenticationInstant="2008-05-14T18:18:47.215Z"⇒
    AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:password">
      <Subject>
        <NameIdentifier NameQualifier=".peoplesoft.com">QEDMO</NameIdentifier>
         <SubjectConfirmation>
           <ConfirmationMethod>urn:oasis:names:tc:SAML:1.0:cm:sender-vouches
           </ConfirmationMethod>
         </SubjectConfirmation>
      </Subject>
   </AuthenticationStatement>
</Assertion>
```

Note these points about PeopleSoft SAML assertions:

- The PeopleSoft SAML token is concerned with the authentication statement only.
- The PeopleSoft SAML token supports SAML with digital signature and encryption. SAML tokens without digital signatures are not supported.
- The PeopleSoft SAML profile of WSS: SOAP Message Security requires that systems support sender-voucher methods of subject confirmation.
- The SAML Assertion validity or condition by default is set to 10 minutes. However, you can override the default time by adding org.apache.ws.security.saml.AssertValidMins=15 in the wssSAML.properties file which is located in the \\WEB-INF\classes\wssSAML.properties directory.

WS-Security SOAP Header

Inbound and outbound transactions that are secured with WS-Security pass the security credentials in a WS-Security SOAP header that is added to the service operation.

The following elements can appear in the WS-Security SOAP header generated by the integration gateway:

Element	*Description*
<wsse:UsernameToken>	Username and optional password to authenticate.
<wsse:Username>	Username to use for authentication. On outbound integrations this name can be generated using the External User ID or Default User ID values that you define on the node definition. In addition, you can select to digitally sign and encrypt this value.
<wsse:Password>	(Optional.) Password for the authentication username. On outbound integrations this password matches that specified for the External User Id or Default User ID used to generate the username. If you select to digitally sign or encrypt the username, this password is digitally signed or encrypted as well.
<saml:Assertion>	SAML assertion token to use for authentication. You can encrypt this value.

The following example shows a WS-Security SOAP header for an outbound service operation generated by the PeopleSoft system:

```
<soapenv:Header>
        <wsse:Security soapenv:mustUnderstand="1" xmlns:wsse=
          "http://docs.oasis-open.org/wss/2004/01/oasis-200401-
            wss-wssecurity-secext-1.0.xsd">
            <wsse:UsernameToken>
                <wsse:Username>PTDMO</wsse:Username>
                <wsse:Password Type="http://docs.oasis-open.org/
                  wss/2004/01/oasis-200401-wss-username-token-profile-
                   1.0#PasswordText">PTDMO</wsse:Password>
            </wsse:UsernameToken>
        </wsse:Security>
    </soapenv:Header>
```

SAML assertions and references to assertion identifiers are contained in the <wsse:Security> element, which in turn is included in the <SOAP-ENV:Header> element. The following example shows SAML assertions conveyed within a WS-Security header as part of a SOAP message:

```
<SOAP-ENV:Envelope>
   <SOAP-ENV:Header>
     <wsse:Security>
       <saml:Assertion> - - - </saml:Assertion>
     </wsse:Security>
   </SOAP-ENV:Header>
  <SOAP-ENV:Body> - - - </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

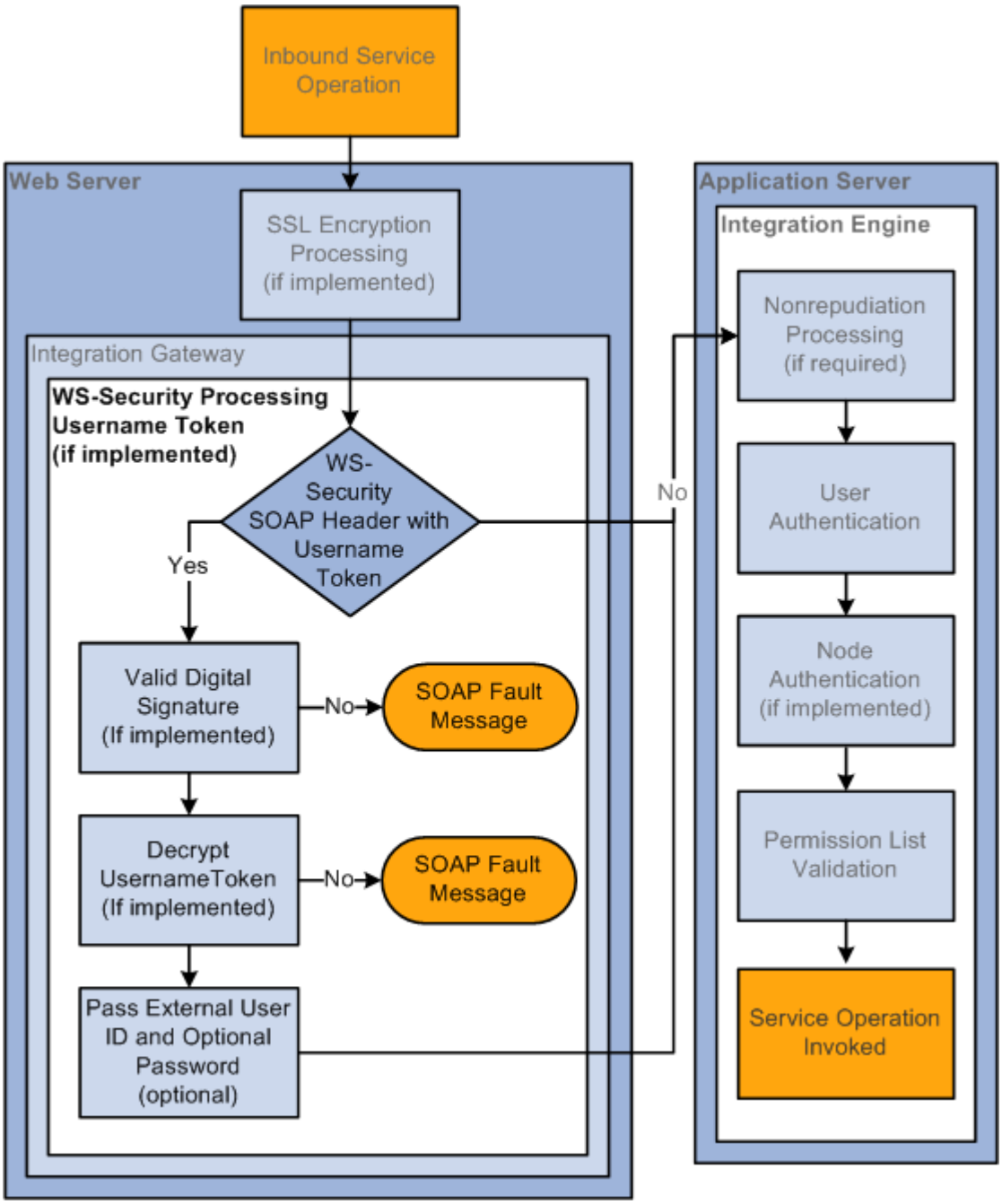
Understanding WS-Security Processing using Username Tokens

This section provides overviews of:

- Inbound WS-Security processing using Username tokens.
- Outbound WS-Security processing using Username tokens.

Inbound WS-Security Processing using Username Tokens

The inbound processing of service operations that are WS-Security-compliant using Username tokens occurs on the integration gateway. The following diagram illustrates inbound WS-Security processing in PeopleSoft Integration Broker:



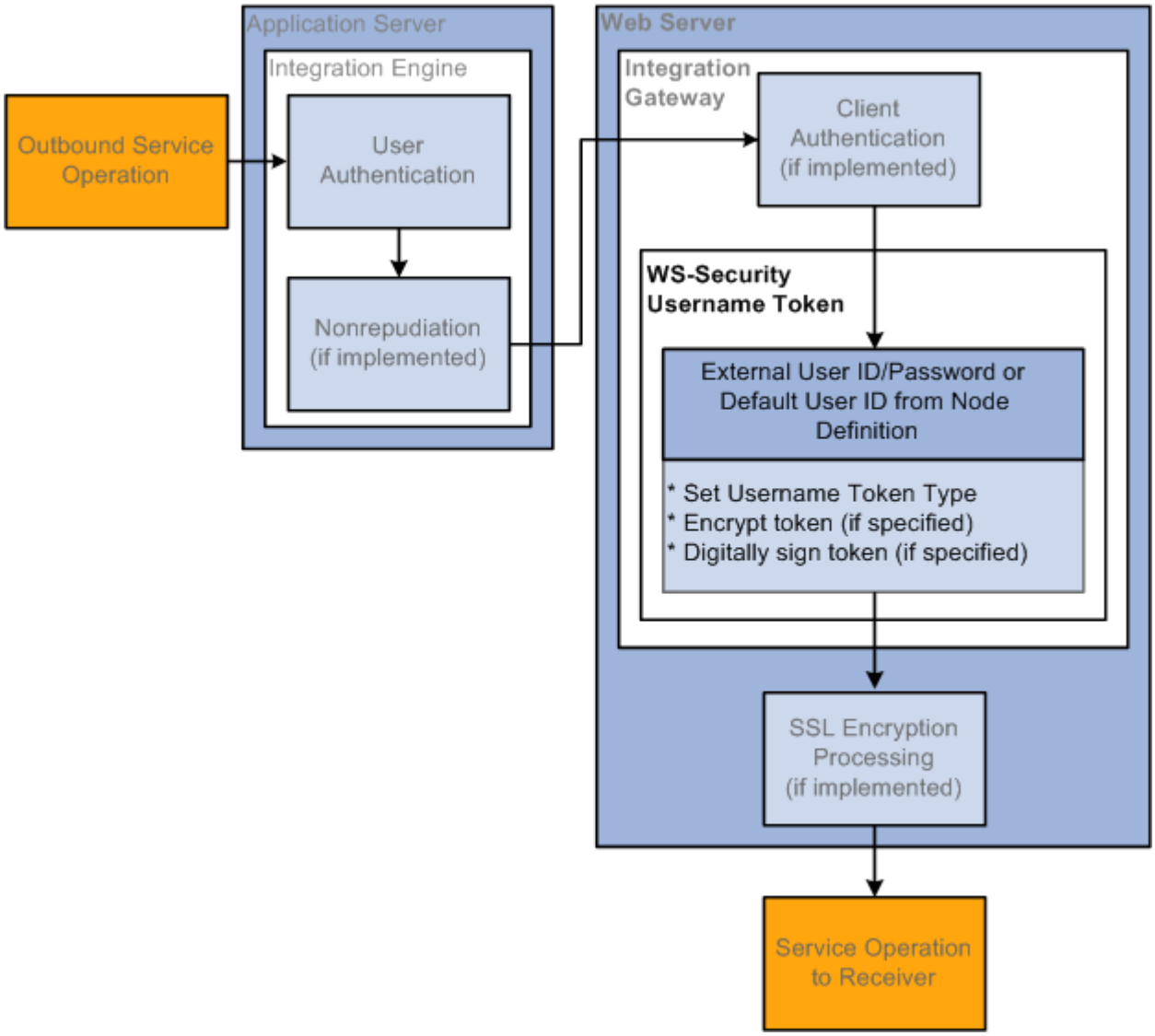
Inbound WS-Security Processing using Username Tokens

When any transaction arrives at the integration gateway, the PeopleSoft system checks for the existence of a WS-Security SOAP header. If it exists, the integration gateway validates the digital signature if it exists, and decrypts the UsernameToken and optional password to restore the user ID information to clear text format.

The integration gateway then passes the user ID information, and UsernameToken password if provided by the sender, to the application server, where additional security processing is performed.

Outbound WS-Security Processing using Username Tokens

The following diagram illustrates WS-Security processing using Username tokens by PeopleSoft Integration Broker on outbound integrations:



Oubtound WS-Security Processing using Username Tokens

When WS-Security is implemented for an outbound service operation, the integration gateway adds a WS-Security SOAP header to the service operation that contains UsernameToken credentials defined in the node definition for the node. The UsernameToken credentials can be comprised of any of the following from the node definition: *External User ID,External Password,* or *Default User ID.* Additionally, you can choose to encrypt and digitally sign the UsernameToken credentials.

See Chapter 11, "Setting Up Secure Integration Environments," Implementing WS-Security for Outbound Integrations (Username and SAML Tokens), page 201 and Chapter 11, "Setting Up Secure Integration Environments," Describing WS-Security Configuration Options for Outbound Integrations (Username Tokens), page 212.

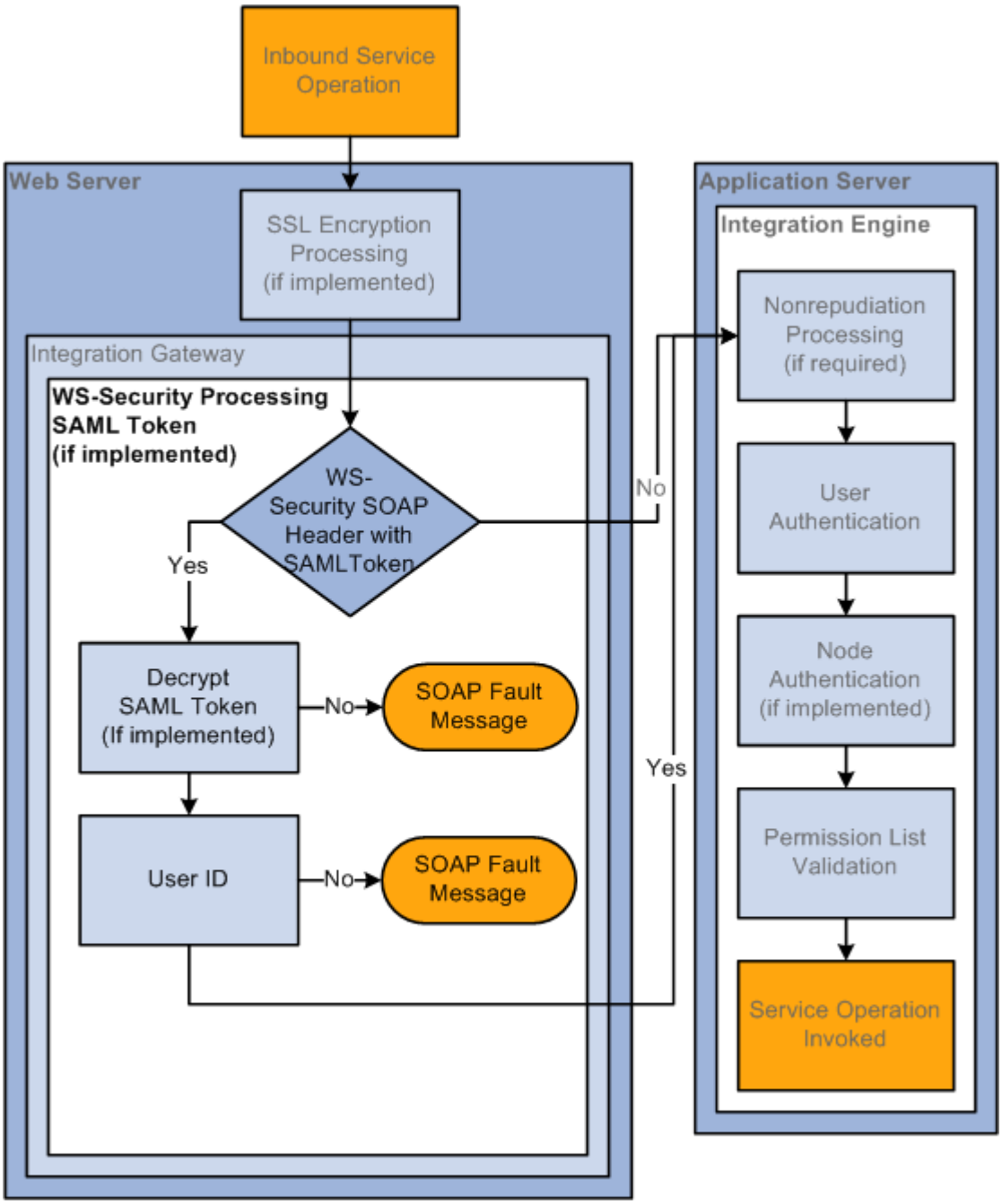
Understanding WS-Security Processing using SAML Tokens

This section provides overviews of:

- Inbound WS-Security processing using SAML tokens.
- Outbound WS-Security processing using SAML tokens.

Inbound WS-Security Processing Using SAML Tokens

The inbound processing of service operations that are WS-Security-compliant using SAML tokens occurs on the integration gateway. The following diagram illustrates inbound WS-Security processing using SAML tokens in PeopleSoft Integration Broker:



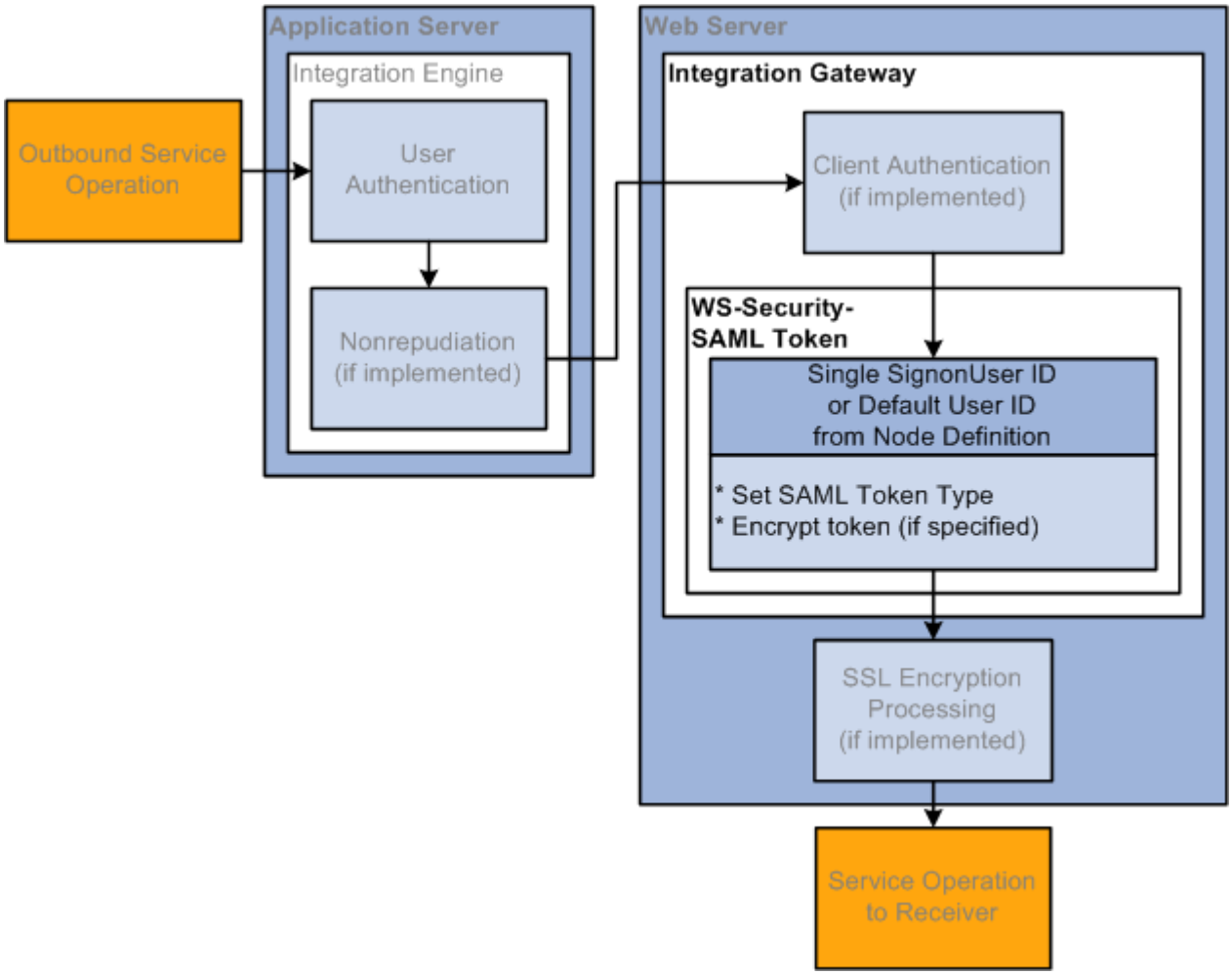
Inbound WS-Security Processing using SAML Tokens

When any transaction arrives at the integration gateway, the PeopleSoft system checks for the existence of a WS-Security SOAP header. If it exists, the integration gateway decrypts the SAML token (if it has been encrypted) to restore the user ID information to clear text format.

The integration gateway then passes the user ID information to the application server, where additional security processing is performed.

Outbound WS-Security Processing Using SAML Tokens

The following diagram illustrates WS-Security processing using Username tokens by PeopleSoft Integration Broker on outbound integrations:



Outbound WS-Security Processing using SAML Tokens

When WS-Security is implemented for an outbound service operation, the integration gateway adds a WS-Security SOAP header to the service operation that contains SAML credentials defined in the node definition for the node. The SAML credentials can be comprised of any of the following from the node definition: *Default User ID* or *PeopleSoft/Single Signon User ID.* Additionally, you can choose to encrypt SAML credentials.

Warning! An any-to-local routing must be used on the sending system for outbound integrations using WS-Security processing and SAML tokens. Integrations will fail using any other routing type.

Prerequisites for Implementing WS-Security in PeopleSoft Integration Broker

To implement WS-Security in PeopleSoft Integration Broker you must install digital certificates.

It is also helpful to set the integration gateway logging to *5* as doing so enables you to see the WS-Security tags in the logs.

See Chapter 11, "Setting Up Secure Integration Environments," Installing Integration Gateway-Based Digital Certificates, page 171 and *Enterprise PeopleTools 8.50 PeopleBook: PeopleSoft Integration Broker*, "Managing Error Handling, Logging, Tracing, and Debugging," Managing Integration Gateway Message and Error Logging.

Implementing WS-Security for Inbound Integrations (Username Tokens)

There is no set up required for implementing WS-Security on inbound integrations. The integration gateway handles all inbound processing.

Implementing WS-Security for Inbound Integrations (SAML Tokens)

This section discusses how to:

- Set up the PeopleSoft system for handling SAML tokens.
- Set up and configure digital certificates.

Setting Up the PeopleSoft System for Handling SAML Tokens

There is some overlap of the steps to set up the PeopleSoft system to handle SAML tokens for integrations using PeopleSoft Integration with those for integrations using WSRP.

The following list describes the steps to set up the PeopleSoft system to handle SAML tokens. Some of the documentation that describes how to perform these steps is located in the *Enterprise PeopleTools 8.50 PeopleBook: PeopleTools Internet Technologies,* as many of the same set-up steps are required to use SAML tokens with WSRP.

- Create the SAML administrator user ID.

  See *Enterprise PeopleTools 8.50 PeopleBook: PeopleTools Portal Technologies*, "Configuring WS-Security For WSRP Consumption and Production," Creating the SAML Adminstrator.

- Set up application server and integration gateway digital certificates.

  See Chapter 11, "Setting Up Secure Integration Environments," Installing Application Server-Based Digital Certificates, page 163 and Chapter 11, "Setting Up Secure Integration Environments," Installing Integration Gateway-Based Digital Certificates, page 171.

- Set SAML assertion data.

  See *Enterprise PeopleTools 8.50 PeopleBook: PeopleTools Portal Technologies*, "Configuring WS-Security For WSRP Consumption and Production," Configuring the SAML Inbound Setup.

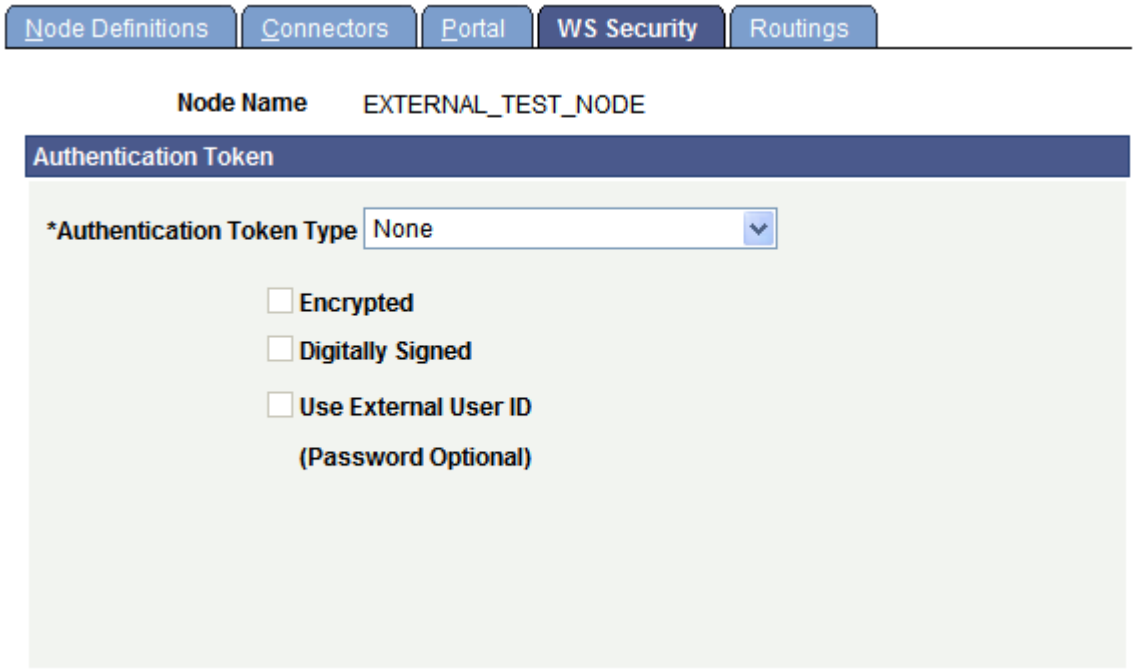
Implementing WS-Security for Outbound Integrations (Username and SAML Tokens)

This section discusses how to:

- Specify an authentication token.
- Specify a default user ID.
- Specify an external user ID and password.

Specifying Authentication Tokens

Use the WS-Security page in the Nodes component (IB_NODE) to set up WS-Security for outbound integrations. The following example shows the page that displays:



Nodes–WS-Security page

The previous example shows the WS Security page that appears by default. The options that appear on this page vary, depending on the authentication token type with which you are working.

To set up WS-Security for Outbound Integrations:

1. Select PeopleTools, Integration Broker, Integration Setup, Nodes.

   The Nodes search page appears.

2. Select the external remote node with which you are integrating.

   The Node Definitions page appears.

3. Click the WS-Security tab.

   The WS-Security page appears.

4. From the Authentication Token Type drop-down list box select an authentication type. The options are:

   - *SAML Token.*
   - *Username Token.*

5. To include additional security options, choose any of the following:

   Additional information about the possible configuration combinations using these options is discussed elsewhere in this section.

   See Chapter 11, "Setting Up Secure Integration Environments," Describing WS-Security Configuration Options for Outbound Integrations (Username Tokens), page 212.

   Encrypt (Optional.) When you check this box, an Encryption Level drop-down list box appears which allows you to choose the level of encryption.

   Digitally Signed This option appears only when you select Username Token.

   (Optional.) Check the box to digitally sign the token information, including the username and password.

   Use Default User ID This option appears only when you select SAML Token.

   (Optional.) Check the box to use the Default User ID specified on the Node Definitions page.

   If this option is not selected the user ID used is the PeopleSoft single signon user ID.

   Use External User ID This option appears only when you select Username Token.

   (Optional.) Check the box to use an external user ID for the username.

   If you select this option, you specify the external user ID and optional password (recommended) on the Node Definitions page.

   Note. If you do not select this option, the Default User ID specified on the Node Definition page is used as the username in the UsernameToken credential.

   See the following Specifying External User IDs and Passwords section.

6. Click the Save button.

7. Click the Node Definitions tab.

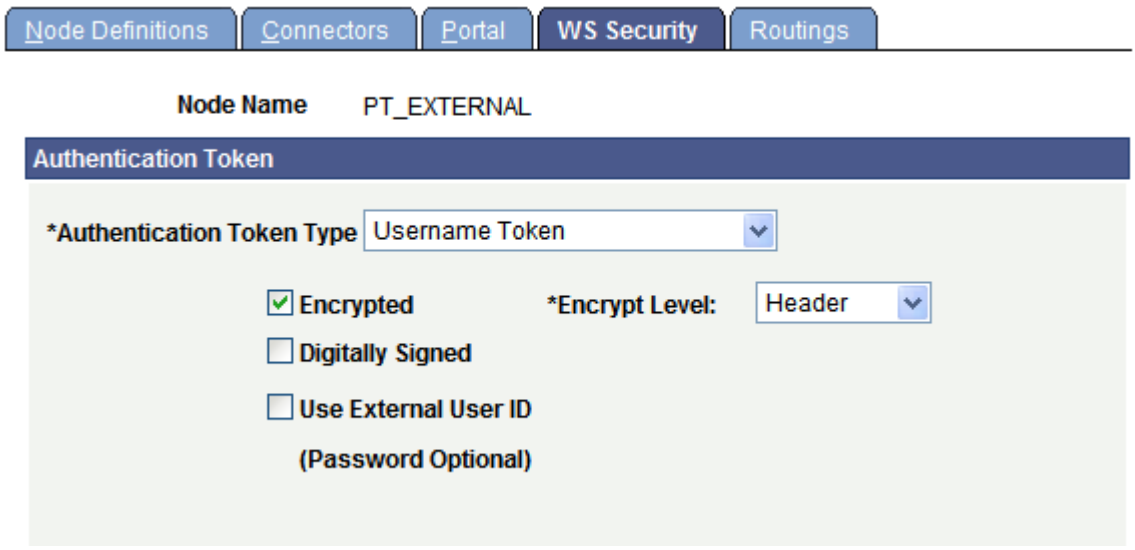
   The Node Definitions page appears.

If you chose to use an external user ID, a dialog box appears indicating that you need to specify the external user ID and optional password. Information on performing that task is described in the Specifying External User IDs and Passwords section.

Encrypting Outbound Messages

When you choose to encrypt messages in an outbound service operation, you have the option to encrypt the entire message, the message body only, or the message header only.

Use the Nodes - WS Security page (IB_NODESECURITY) to work with any of these options. The following example shows the page:



Choosing a message encryption level

When working with an external node type and you select the Encrypt box, the Nodes — WS Security page displays an Encrypt Level drop-down list box from which you can choose an encryption level.

The encryption level options are:

- *All*. This option encrypts the entire message including the message header and body.
- *Body*. Encrypts the message body only.
- *Header*. (Default) Encrypts the message header only.

Specifying Default User IDs

When using the SAML token type to implement WS-Security, you have the option to use the default user ID for processing. You set the default user ID on the external remote node definition using the Node Definitions page.

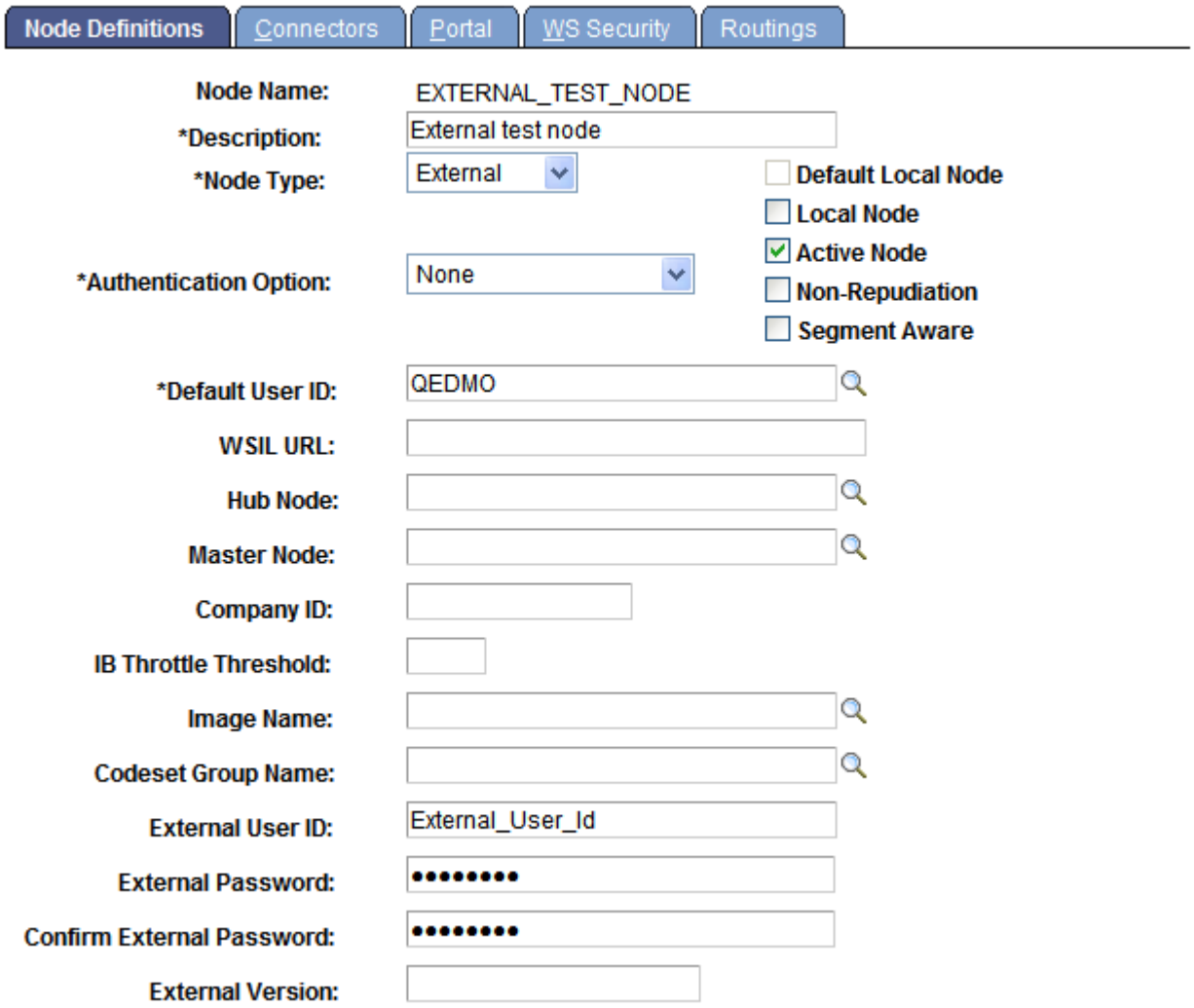
When you create a node definition, you must supply a value for the Default User ID. However, you a free to change the value at any time.

See Chapter 7, "Adding and Configuring Nodes," Configuring Nodes, page 130.

If you do not check the Default User ID option, the system uses the PeopleSoft User ID/Single Signon User ID for the transaction.

Specifying External User IDs and Passwords

When using the Username token type to implement WS-Security, you have the option to specify an External User ID. You define the external user ID on the Node Definitions page shown in the following example:



Specifying an external user ID and password

When specifying an external user ID, specifying an external user ID password is recommended.

Note. The Confirm External Password field appears after you specify the external password and tab out of the field.

To specify the External User ID and Password:

1. On the Node Definitions page, in the External User ID field, enter an external user ID.
2. (Optional.) In the External Password field, enter the password for the external user ID.

   Tab out of the field. A Confirm External Password field appears.
3. In the Confirm External Password field, re-enter the external user ID password.
4. Click the Save button.

Development Considerations for Implementing WS-Security in Asynchronous Request/Response Service Operations

This section discusses development considerations for implementing WS-Security in asynchronous request/response service operations and discusses how to:

- Digitally sign responses in asynchronous request/response service operations.
- Secure responses in asynchronous request/response service operations.

Digitally Signing Responses in Asynchronous Request/Response Service Operations

This section applies to inbound asynchronous request/response service operations defined with any-to-local routing definitions.

In any-to-local routing definitions, no requesting node is present. As a result no digital certificate information that is normally defined at the node level is included with the request. However, the request does contain a RequestAliasName parameter that is populated with the certificate issuer's credentials that the integration gateway uses to process the request.

To digitally sign a response for an asynchronous request/response service operation, in the response set the RequestAliasName parameter equal to the same value that was set for the parameter on the request message. The integration gateway reads that value and can then determine the certificate to use in the response.

The following example shows how to code a digitally-signed response for an asynchronous request/response service operation:


```
import PS_PT:Integration:INotificationHandler;

class FLIGHTDATA_RETURN implements PS_PT:Integration:INotificationHandler
   method FLIGHTDATA_RETURN();
   method OnNotify(&MSG As Message);
end-class;

/* constructor */
method FLIGHTDATA_RETURN
end-method;

method OnNotify
   /+ &MSG as Message +/
   /+ Extends/implements PS_PT:Integration:INotificationHandler.OnNotify +/
   /* Variable Declaration */


   Local string &str, &value;
   Local Rowset &rs;
   Local integer &num;

   Local Message &MSG_resp;
   Local Record &FLIGHTDATA, &REC;

   &rs = &MSG.GetPartRowset(1);
   /* process request data */

   &MSG_resp = CreateMessage(Operation.FLIGHTPLAN_ARR, %IntBroker_Response);
   &rs = &MSG_resp.GetPartRowset(1);
   /* popualate response data */

   If &MSG.IsSourceNodeExternal Then

      /* set WS Addressing information and WS_RequestAliasName
   if security to be added to response message */

      &MSG_resp.IBInfo.WSA_MessageID = &MSG.IBInfo.WSA_MessageID;
      &MSG_resp.IBInfo.WSA_ReplyTo = &MSG.IBInfo.WSA_ReplyTo;
      &MSG_resp.IBInfo.WS_RequestAliasName = &MSG.IBInfo.WS_RequestAliasName;

   Else
      /* request from PSFT system */
      &MSG_resp.IBInfo.WSA_ReplyTo = &MSG.TransactionId;

   End-If;

   %IntBroker.Publish(&MSG_resp);
end-method;
```

Securing Responses in Asynchronous Request/Response Service Operations

PeopleSoft Integration Broker sends responses for asynchronous request/response service operations to the URL set in the Target Location field in on the Service Configuration page. The URL specified on this page is typically not secure, as it is the URL used for all WSDL, schemas, and web transactions.

When providing asynchronous request/responses, you can dynamically override the URL using the IBInfo object property WSA_ReplyTo. For example:

```
&MSG.IBINFO.WSA_ReplyTo
```


You set this property typically before the publish or during the OnSend event.

Overriding Node-Level WS-Security Settings on Routing Definitions

This section discusses how to:

- Override WS-Security settings on routing definitions (General).
- Override WS-Security settings on routing definitions (Synchronous Responses).
- Override WS-Security settings on routing definitions (Asynchronous Requests/Responses).

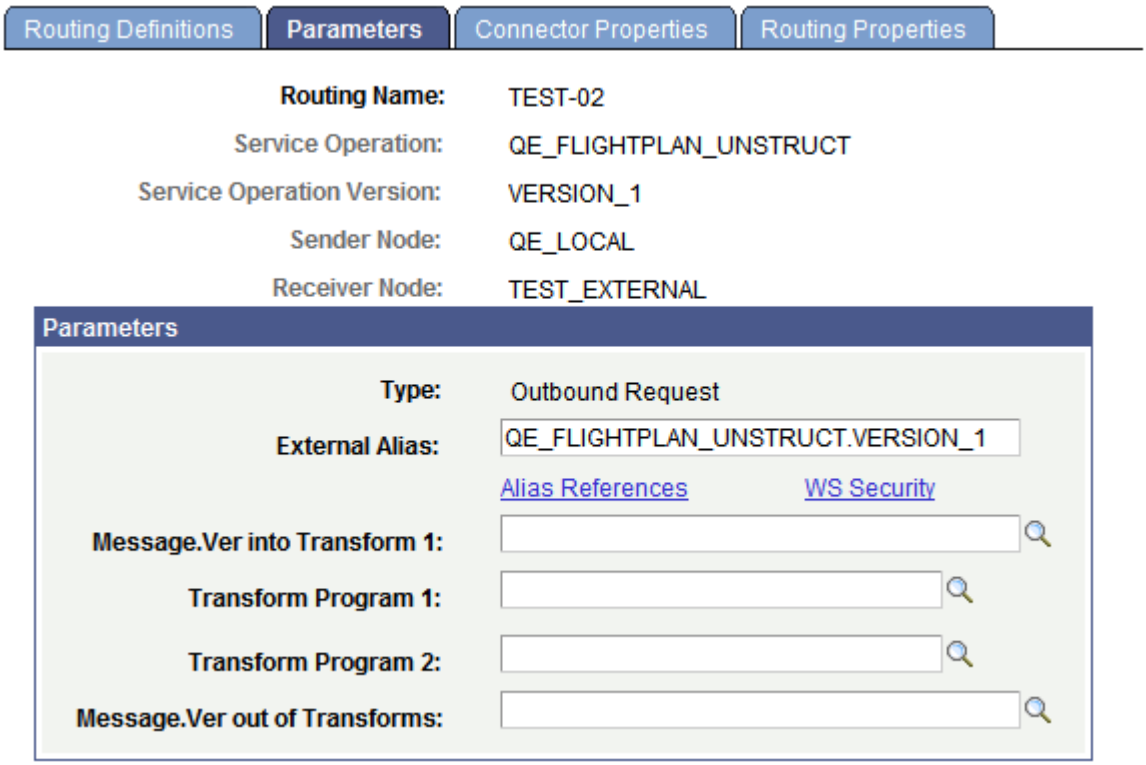
Understanding Overriding Node-Level WS-Security Settings on Routing Definitions

You can override node-level WS-Security settings on individual routing definitions for outbound request and response messages. The security settings that you can override are the same as those that appear on the Nodes-WS Security page.

When overriding WS-Security settings for synchronous request messages and asynchronous request/response messages, there are PeopleCode considerations of which you should be aware. In addition, the outbound security overrides on the routing definition for these transaction types work in concert with inbound security validation set on the service operation. These considerations and options are discussed in this section.

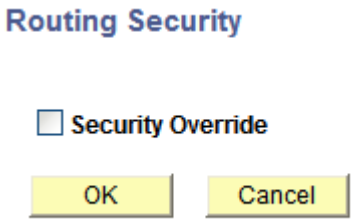
Overriding WS-Security Settings on Routing Definitions (General)

When you are working with an outbound routing definition to a external node, the Routing Definitions-Parameters page features a WS-Security link, as shown in the following example:



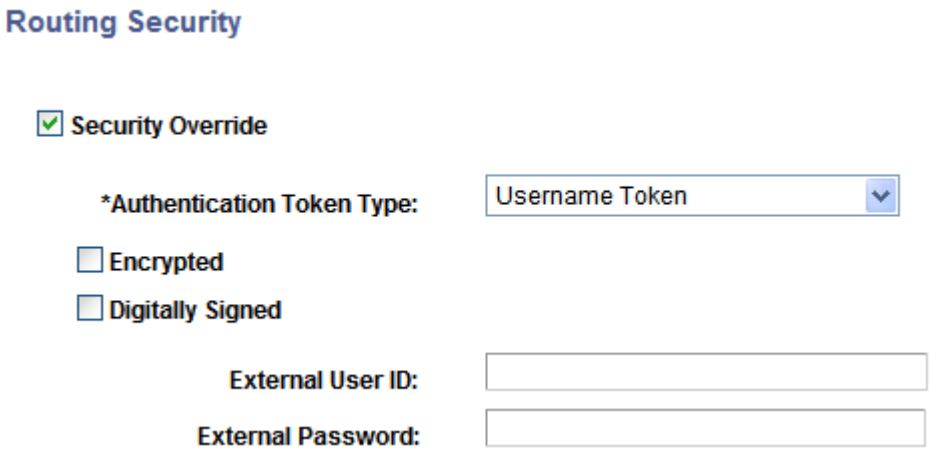
The WS Security link to override node-level WS-Security settings appears under the External Alias field.

When you click the WS Security link the Routing Security page (IB_ROUTINGDEFN_SEC) appears as shown in the following example:



The Routing Security does not display any options to override until you check the Security Override box.

When you check the Security Override box on the Routing Security page, the WS-Security options that you can override appear on the page, as shown in the following example:



Routing Security page showing options you can override when Username Token is the authentication type.

The WS-Security options that appear and that you can set and override in a routing definition, depend on the authentication type and encryption option, if any, set.

To override WS-Security options on a routing definition:

1. Select PeopleTools, Integration Broker, Integration Setup, Routings, and select or add a routing definition.
2. Click the Parameters tab.

   A WS Security link appears for the outbound request or response message, depending on whether the external node is the sending or receiving node.
3. Click the WS Security link.

   The Routing Security page appears.
4. Select the Security Override box.

   The Authentication Token Type drop-down list box appears.
5. From the Authentication Token Type drop-down list box, choose an authentication token type with which to work. The options are:

   - *None.* (Default.)
   - *SAML Token.*
   - *Username Token.*

If you choose SAML token or Username token, additional security options appear with which to work. SAML token and Username token security options for outbound messages are discussed elsewhere in this chapter.

See Chapter 11, "Setting Up Secure Integration Environments," Implementing WS-Security for Outbound Integrations (Username and SAML Tokens), page 201.

Overriding WS-Security Options on Routing Definitions (Synchronous Responses)

To override the security for a synchronous response, on the Routing Definitions-Parmaters page, select the WS Security link on the Outbound Response Parameter:

If the Encrypted check box is selected then the response message sent from the consumer must be digitally signed if the routing is an any-to-local type routing.

In addition, you can reject a request message that is not digitally signed. To do so, on the Service Operations-General page, from the Security Verification drop-down list select Digitally Signed.

Overriding WS-Security Options on Routing Definitions (Asynchronous Request/ Response)

You can use the Routing Security page to encrypt and/or digitally sign outbound responses in asynchronous request/response transactions.

If you select the Encrypted box for the outbound response message, then the message sent from the consumer must be digitally signed if the routing is an any-to-local type routing.

You can then select the *Digitally Signed* option for Security Verification on the service operation to reject a non-signed request message.

To successfully use WS-Security on a response, within the PeopleCode the RequestAliasName must also be populated on the response Message object from the request Message object. Here is an example in PeopleCode:

```
&MSG_resp = CreateMessage(Operation.FLIGHTPLAN_DOC_ARR, %IntBroker_Response);
&MSG_resp.IBInfo.WSA_MessageID = &MSG.IBInfo.WSA_MessageID;
&MSG_resp.IBInfo.WSA_ReplyTo = &MSG.IBInfo.WSA_ReplyTo;
&MSG_resp.IBInfo.WS_RequestAliasName = &MSG.IBInfo.WS_RequestAliasName;
```

The system validates proper encryption based on the request verification selected on the service operation. Select the desired security validation. If the security on the actual request message does not match the value specified in the Security Verification field, an error is sent back to the client.

Implementing WS-Security on Services Consumed Using the Consume Web Service Wizard

You can implement WS-Security on service operations you consume using the Consume Web Service Wizard.

When using the Consume Services wizard, there is an option to use the default pre-defined *WSDL_NODE* node or use another existing node as the receiving node for the consumed service operations. The action to take to implement WS-Security on consumed services depends on which of these nodes you are using.

Using the Default WSDL Node

If you choose the default *WSDL_NODE* node as the receiving node, then you should add/override the node-level WS-Security settings by using the Routing Security page on the routing definition for the created service operation.

If you are using the *WSDL_NODE* node as the receiving node and the message is to be encrypted, the WS_RequestAliasName must be populated on the request message with the alias name used when adding the provider certificate to the gateway keystore. In addition, in PeopleCode after the message is created add this alias as follows:

```
&MSG.IBinfo.WS_RequestAliasName = "the alias name from the gateway keystore";
```

Using an Existing Node

If you use a node other than the *WSDL_NODE* as the receiving node then you can specify the security settings at the node level on the Nodes- WS Security page or on the Routing Security page on the routing definition.

Describing WS-Security Configuration Options for Outbound Integrations (Username Tokens)

This section discusses:

- Recommended WS-Security configurations for outbound integrations.
- Supported WS-Security configurations for outbound integrations.
- Non-secure WS-Security configurations for outbound integrations.

Recommended WS-Security Configurations for Outbound Integrations (Username Token)

The following table highlights recommended WS-Security configurations on the PeopleSoft system for outbound integrations using Username tokens. Note that the configuration is always performed on the remote node and that the node type is always *External*.

External User ID and Password	*Authentication Type*	*With SSL Encryption*	*Results*
Both	Username Token with the External User ID option.	Yes	The system uses the external user ID and password to generate the username token. The token is generated in clear text.
Both	Username Token with the following other options: • External User ID. • Encrypted. • Digitally signed.	No	The system uses the external user ID and password to generate the username token. The token is encrypted and digitally signed.
Both	Username Token with the Digitally signed option.	Yes	The system uses the external user ID and password to generate the username token. The token is digitally signed.

External User ID and Password	Authentication Type	With SSL Encryption	Results
External User ID only.	Username Token with the External User ID option.	Yes	The system uses the external user ID to generate the username token. The token is generated in clear text.
External User ID only.	Username Token with the following other options: • External User ID. • Encrypted. • Digitally signed.	No	The system uses the external user ID to generate the username token. The token is encrypted and digitally signed.
External User ID only.	Username Token with the following other options: • External User ID. • Digitally signed.	No	The system uses the external user ID to generate the username token. The token is digitally signed.

Supported WS-Security Configurations for Outbound Integrations (Username Token)

The following table highlights supported WS-Security configurations on the PeopleSoft system for outbound integrations using Username tokens. Note that the configuration is always performed on the remote node and that the node type is always *External*.

External User ID and Password	Authentication Type	With SSL Encryption	Results
None.	Username Token option only.	Yes.	The system uses the default user ID defined on the node definition to generate the username token. The token is generated in clear text.
None.	Username Token with the following other options: • Encrypted. • Digitally signed.	No.	The system uses the default user ID defined on the node definition to generate the username token. The token is encrypted and digitally signed.
None	Username Token with the Digitally Signed option.	No.	The system uses the default user ID defined on the node definition to generate the username token. The token is digitally signed.

Non-Secure WS-Security Configurations for Outbound Integrations (Username Token)

The following table highlights non-secure WS-Security configurations on the PeopleSoft system for outbound integrations using Username tokens. Note that the configuration is always performed on the remote node and that the node type is always *External*.

Warning! The following configurations are not secure! This information is provided to advise you about configurations that can lead to breaches in security. Use the recommended or supported configurations discussed in the previous sections for configuring your system .

External User ID and Password	*Authentication Type*	*With SSL Encryption*	*Results*
Both	Username Token with the External user ID option.	No.	The system uses the external user ID and password to generate the username token. The token is generated in clear text.
None.	Username Token option only.	No.	The system uses the default user ID defined on the node definition to generate the username token. The token is generated in clear text.
Both	Username Token with the following options: • External user ID. • Encrypted.	No.	The system uses the external user ID and password to generate the username token. The token is encrypted.
None.	Username Token with the following options: • External user ID. • Encrypted.	No.	The system uses the default user ID and password to generate the username token. The token generated is encrypted.

WS-Security SOAP Header Examples (Username Token)

This section provides the following WS-Security code examples:

- WS-Security UsernameToken in ciphertext and digitally signed.
- WS-Security UsernameToken with clear text user name and password.
- WS-Security UsernameToken with clear text with user name only.

Example 1: WS-Security UsernameToken in Ciphertext and Digitally Signed

The following code example shows a WS-Security SOAP header that contains a UsernameToken in cipher text and that is digitally signed. This is the most secure configuration for WS-Security in PeopleSoft Integration Broker.

```
<soapenv:Header>
   <wsse:Security soapenv:mustUnderstand="1" xmlns:wsse="http://docs.
     oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0
      .xsd">
      <xenc:EncryptedKey>
        <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/
           xmlenc#rsa-1_5"/>
         <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
            <wsse:SecurityTokenReference>
               <ds:X509Data>
                  <ds:X509IssuerSerial>
                     <ds:X509IssuerName>CN=PeopleTools TEST root CA,
                       DC=peoplesoft,DC=com,OU=PeopleTools Development,
                       O=PeopleSoft Inc,L=Pleasanton,ST=CA,C=US</ds:
                       X509IssuerName>
                     <ds:X509SerialNumber>174697022083003580418117</ds:
                       X509SerialNumber>
                  </ds:X509IssuerSerial>
               </ds:X509Data>
            </wsse:SecurityTokenReference>
         </ds:KeyInfo>
         <xenc:CipherData>
            <xenc:CipherValue>q8ytyn0kRisc3i7GwGtoQuU6NSXfvSNoJg76PWpppt
             4b4DoH8bRObvht8GLu904OExYBrNDB26qqOlKVpIzGrCJFgetlhikGghH/u2
             9GC96+YfFdxSFqcJo5PpJR1KnVZP0sKO4IHVIEcuxp7MonoV6dm5kd0d8atVw
             KXhJe5Yk=</xenc:CipherValue>
         </xenc:CipherData>
         <xenc:ReferenceList>
            <xenc:DataReference URI="#EncDataId-13925529"/>
         </xenc:ReferenceList>
      </xenc:EncryptedKey>
      <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
         <ds:SignedInfo>
            <ds:CanonicalizationMethod Algorithm="http://www.w3.org/
             2001/10/xml-exc-c14n#"/>
            <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/
             xmldsig#rsa-sha1"/>
            <ds:Reference URI="#id-763474">
               <ds:Transforms>
                  <ds:Transform Algorithm="http://www.w3.org/2001/10/
                   xml-exc-c14n#"/>
               </ds:Transforms>
               <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/
                 xmldsig#sha1"/>
               <ds:DigestValue>cNBCuvnSP5MMlsJvaHMrZm9CsK0=</ds:
                 DigestValue>
            </ds:Reference>
            <ds:Reference URI="#id-13925529">
               <ds:Transforms>
                  <ds:Transform Algorithm="http://www.w3.org/2001/10/
                   xml-exc-c14n#"/>
               </ds:Transforms>
               <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/
                 xmldsig#sha1"/>
               <ds:DigestValue>p+IodojBA2QzX6p9xe6PKJyUKSg=</ds:
                 DigestValue>
            </ds:Reference>
         </ds:SignedInfo>
         <ds:SignatureValue>D/kTMJZvxnv7fjWzmvKC1xe8VSDiSz4lZDzFrf8q
           FFoXux+C2xD47TLWnD7m8ejp/Un3mzjWkVN8S4FpwRr/ymrxWTKWLrjCO
           zmjSW+ZbjGvs5UfpFyzEH7PWrXt+LnTeMKKJWYjzOi7HCHCVK9aC/RZCt
           7PkCbSZ7DJoOQO/lU=
         </ds:SignatureValue>
         <ds:KeyInfo Id="KeyId-28705465">
```

```
                <wsse:SecurityTokenReference wsu:Id="STRId-7131385" xmlns:wsu=
                  "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
                   wssecurity-utility-1.0.xsd">
                   <ds:X509Data>
                      <ds:X509IssuerSerial>
                         <ds:X509IssuerName>CN=PeopleTools TEST root CA,DC=
                           peoplesoft,DC=com,OU=PeopleTools Development,
                            O=PeopleSoft Inc,L=Pleasanton,ST=CA,C=US
                          </ds:X509IssuerName>
                         <ds:X509SerialNumber>17433215564084276520762O
                         </ds:X509SerialNumber>
                      </ds:X509IssuerSerial>
                   </ds:X509Data>
                </wsse:SecurityTokenReference>
             </ds:KeyInfo>
          </ds:Signature>
          <xenc:EncryptedData Id="EncDataId-13925529" Type="http://www.w3.
            org/2001/04/xmlenc#Element">
             <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/
               xmlenc#tripledes-cbc"/>
             <xenc:CipherData>
                <xenc:CipherValue>wqrOr/efBcghEdcTPZMPqbrUu9mF+iCSLf2UhLYjOc
                 Vg30+58TX3FCKXJhExi3iEdbuVrYt60mq3Maka6cg6+0JXw0Qmbjbl5qG8p
                 sHajRtenvZc3dJeLRDclplbqUw65cDvBqQz+3+K5DBMh+tIlutf+0j3D9MiO
                 3ht4Ni4bJ9Zk/h+DY9y05px2xtOMsXSrEhn4STGz4SdaOwFYHDUTT+y+D6zj
                 GYxpRAexVQxAkjehW1JEGhyaqqdDmIYPJxCSy8JBc7xL/CaUng98ak8hAr38I
                 obBt1qjlYjGo9VybfrX5j9lqn6pcrWX6x3o/9JYXeiaY36qHj+jVm0STq1fPr
                 DDfh6ZI0/aeks83MnesMrX9bB7aKOo67DPjJstRvW/qfbIo3wYgv+3Jl68sHv
                 u6p6GZEujaLIYIosJ+HtDzmZ2Q9aOtkk7+zFwDohkljAwmNSe3bt9e2i60pgF
                 fVYcxg1Pwfz03MyKm83m5cLT9INb8LHK/GsKOl+9GvQ49nsJ6EYuAcPO4Q8Sr
                 BvLVVPY3Qljw+4ZOZOEcndxVw+vU9n7cAMyeYa7p5Jpl6l2naeC4J98MIa16D
                 CuVdvLIkipurkn2lbVYe5/m0SYbVibvTWE3BIQlWzF/mRHKkOhBhTaKg/Y/Q7
                 sRlKcxKHtjnsjX2d4hTqTRYOoKFEH5sVi+gtyhgogiXRjg8wCAS68cYVwAFre
                 W9xf2/ojGJFcO354Sk5rWt3GZzK8yRG5Jcgf5pgxnKC3LVgvvGPQM2Q/yGy1N
                 OrXDhtzc80zM2SIOjv3A90Gzj9RGKzrWm+bw4QlhveY+rwyZGZRu3ibVUm+mi
                 Ul7CdBBbrLOfz9xY45w3H2c6mtu98OwhuoiYHeVS/FkdpL+ztLmZi7gINIAQi
                 sCZudpyKsZIcEhTPbTjQcdCVPZim1v9HFft00cSOE1u1CVEYNOSuCisrLJIch
                 zAtE7gfa86/NcyEGmUBtvRsGVPkPq81cw1AosV8x4+KPCpTjxxeuMKGrowC2h
                 Y/7DY+IYn4
                </xenc:CipherValue>
             </xenc:CipherData>
          </xenc:EncryptedData>
       </wsse:Security>
    </soapenv:Header>
```

Example 2: WS-Security UsernameToken with Clear Text Username and Password

The following code example shows a WS-Security SOAP header that contains a clear-text UsernameToken credential that contains a user name and a password.

```
<soapenv:Header>
  <wsse:Security soapenv:mustUnderstand="1" xmlns:wsse="http://docs.oasis-
    open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
     <wsse:UsernameToken>
         <wsse:Username>IBUSER</wsse:Username>
         <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/
           oasis-200401-wss-username-token-profile-1.0#PasswordText">IBUSER1
         </wsse:Password>
     </wsse:UsernameToken>
  </wsse:Security>
</soapenv:Header>
```


Example 3: WS-Security UsernameToken with Clear Text User Name Only

The following code example shows a WS-Security SOAP header that contains a clear-text UsernameToken credential that contains only a user name. No password is specified.

```
<soapenv:Header>
  <wsse:Security soapenv:mustUnderstand="1" xmlns:wsse="http://docs.
    oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
     <wsse:UsernameToken>
        <wsse:Username>QEMGR</wsse:Username>
        <wsse:Password/>
     </wsse:UsernameToken>
  </wsse:Security>
</soapenv:Header>
```

Implementing Client Authentication

This section provides an overview of client SSL authentication, prerequisites, and discusses how to implement client authentication.

Understanding Client Authentication

As mentioned previously in this chapter, outbound transactions connect from the PeopleSoft application server to the integration gateway using an HTTP over MIME connection. Client SSL encryption allows you to secure this connection. Client SSL encryption is not implemented on inbound transactions from the integration gateway to the application server, since this connection is made using a Jolt connection.

Client SSL encryption is typically implemented when the application server and integration gateway each reside on separate machines.

Client SSL encryption is implemented using digital certificates and you must have them installed on the integration gateway.

Note. You must have web server SSL encryption set up and implemented to use client SSL authentication. With web server SSL set up and implemented, client SSL authentication will fail.

After digital certificates are installed, there are no other steps required to implement client SSL authentication.

See Also

Chapter 11, "Setting Up Secure Integration Environments," Installing Integration Gateway-Based Digital Certificates, page 171

Implementing Nonrepudiation

This section provides and overview of nonrepudiation and discusses how to configure nonrepudiation.

Understanding Nonrepudiation

PeopleSoft Integration Broker applies nonrepudiation to cross-node messaging by digitally signing service operation requests and their responses.

Nonrepudiation Processing Overview

In PeopleSoft applications, nonrepudiation provides two-way protection; both the request and its response are nonrepudiated. PeopleSoft Integration Broker uses PKI technology to implement nonrepudiation for integrations. Each participating node's keystore contains its own private key and the public keys of the nodes with which it exchanges nonrepudiation service operations.

Nonrepudiation works in the following manner:

1. Node A generates a number, known as a *digest,* which uniquely identifies its service operation request.
2. Node A uses its private key to generate a signature based on the digest, and inserts the signature into the nonrepudiation service operation request.
3. Node A sends the nonrepudiation request to Node B.
4. When it receives the nonrepudiation request, Node B uses Node A's public key in its keystore to confirm the integrity of the digest.

   It then separately recreates the digest from the service operation, and compares it to the received digest to confirm the integrity of the service operation.
5. Node B generates a digest that uniquely identifies its response.
6. Node B uses its private key to generate a signature based on the digest, and it inserts the signature into the nonrepudiation response to confirm receipt of the nonrepudiation request.
7. Node B sends the nonrepudiation response to Node A.
8. When the nonrepudiation response is received, Node A uses Node B's public key in its keystore to confirm the integrity of the digest.

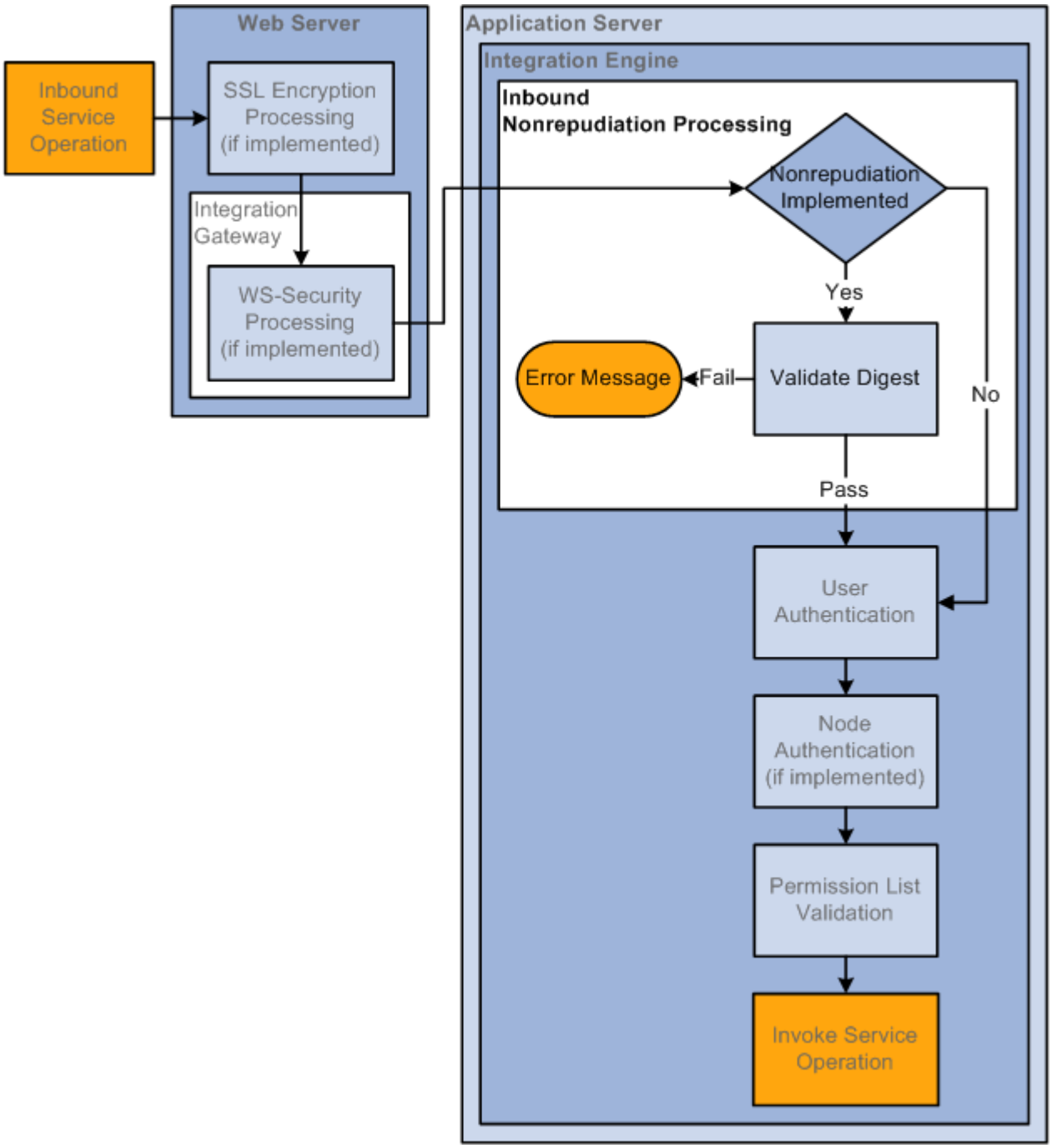
   It then separately re-creates the digest from the service operation and compares it to the received digest to confirm the integrity of the service operation content.

Nonrepudiation produces the following results:

- The sending node cannot repudiate that the service operation was sent, because the receiving node has a copy of the request signed by the sender.
- The receiving node cannot repudiate that the service operation was received and processed, because the sending node has a copy of the response signed by the receiver.
- The service operation integrity is verified, because the validated signature of each nonrepudiated service operation assures that the service operation content as received, exactly matches the content as sent.

Inbound Nonrepudiation Processing

The following diagram illustrates inbound nonrepudiation processing:



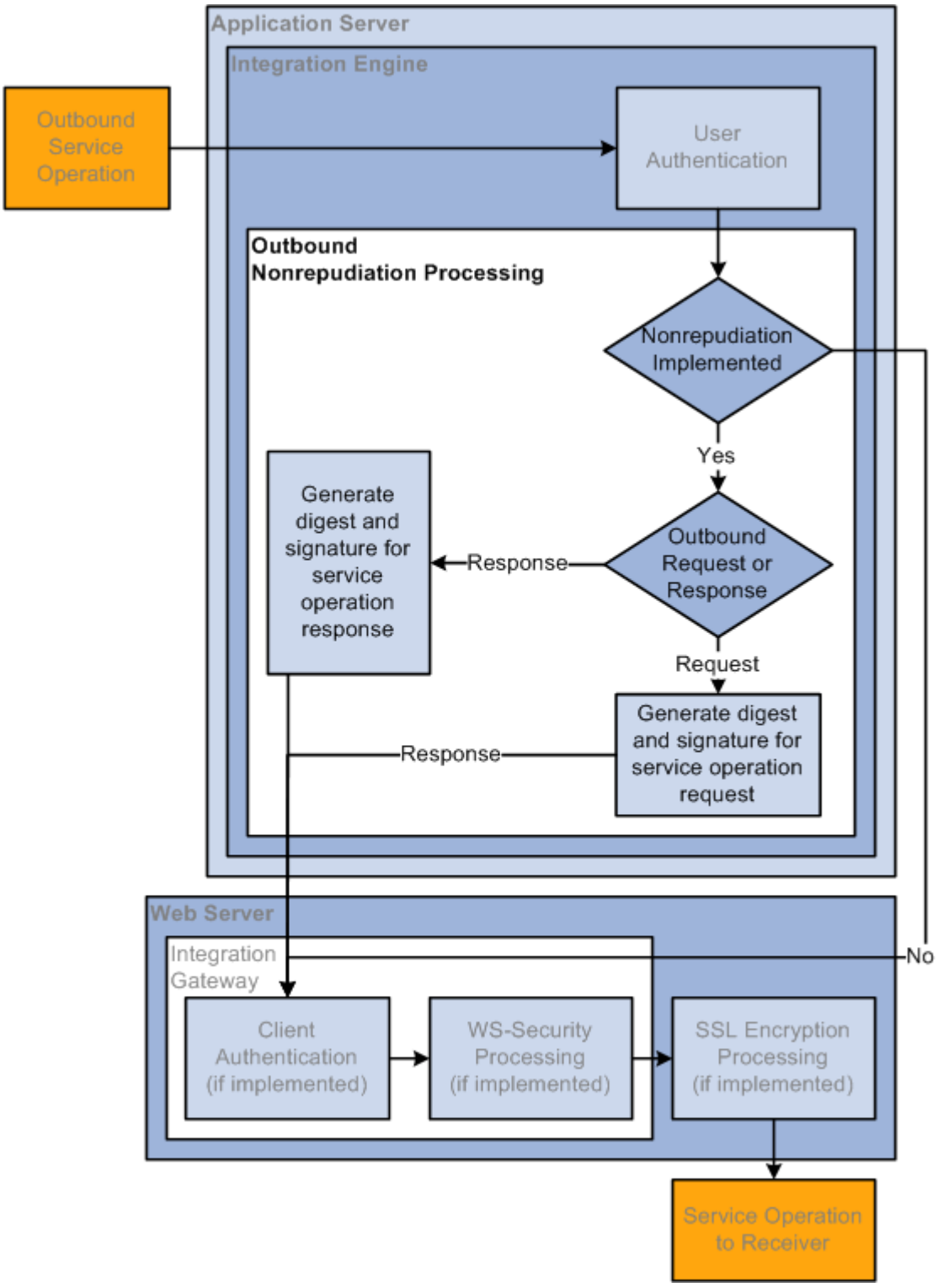
Inbound Nonrepudiation Processing

In inbound nonrepudiation processing, the system uses the integration partner's public key to validate the digest attached to the inbound service operation. It then uses its private key to recreate the digest on the service operation to validate the integrity of the service operation content.

If the system is able to validate the integrity of the digest and the service operation content, the service operation then goes through the user authentication process. If the system is unable to validate the digest or the service operation content, the transaction fails.

Outbound Nonrepudiation Processing

The following diagram illustrates outbound nonrepudiation processing:



Outbound Nonrepudiation Processing

On outbound service operations, the system determines if the service operation is a request or a response.

When the service operation is a request, the system uses its private key to generate a digest and signature, and attaches those items to the request.

When the service operation is an outbound response, the system uses its private key to generate a signature and response and inserts them into the service operation.

Prerequisites for Implementing Nonrepudiation

You must install application server-based digital certificates on both sending and target systems to implement nonrepudiation.

See Chapter 11, "Setting Up Secure Integration Environments," Installing Application Server-Based Digital Certificates, page 165.

Configuring Nonrepudiation

This section discusses nonrepudiation configuration tasks on sending and target PeopleSoft systems using PeopleSoft Integration Broker.

If a participating node doesn't use PeopleSoft Integration Broker, that node is still responsible for managing the appropriate private and public keys, inserting properly formatted signatures in the nonrepudiation service operation it sends, and properly handling signatures in the service operations that it receives.

With both archived and active nonrepudiation service operations, you can regenerate the digest in the Service Operations Monitor to reconfirm that it matches the attached digest.

See *Enterprise PeopleTools 8.50 PeopleBook: Integration Broker Service Operations Monitor*, "Viewing Service Operation Nonrepudiation Signature Information."

Configuring Nonrepudiation on Sending PeopleSoft Systems

This section discusses configuring nonrepudiation on sending systems for asynchronous or synchronous transactions.

Prerequisites for configuring nonrepudiation are discussed elsewhere in this section.

See Chapter 11, "Setting Up Secure Integration Environments," Prerequisites for Implementing Nonrepudiation, page 221.

To configure nonrepudiation for service operations on the source system you must:

- Select the Non-Repudiation check box on the service operation that will be invoked.
- Select the Non-Repudiation check box on the remote node definition that represents the target system.

Configuring Nonrepudiation on Target PeopleSoft Systems

You must supply the digital certificates containing the private and public keys required for nonrepudiation transactions.

No additional configuration is required on target PeopleSoft systems to handle nonrepudiated service operations. A nonrepudiated service operation received by a target PeopleSoft system will attempt to validate the service operation regardless if the local node and service operation are set for nonrepudiation.

Saving Nonrepudiated Service Operations

To save nonrepudiation service operations for future reference, you must archive them.

See *Enterprise PeopleTools 8.50 PeopleBook: Integration Broker Service Operations Monitor*, "Archiving Service Operation Instances."

Managing User Authentication

This section provides overviews of user authentication, outbound user authentication, inbound user authentication, and discusses how to:

- Activate user authentication on service operations.
- Set up user authentication on sending systems.
- Exclude Peoplesoft authentication tokens in outbound requests to PeopleSoft nodes.

Understanding User Authentication

In PeopleTools 8.48 and later releases, access to invoke service operations is enforced at the user level.

When integrating with other PeopleSoft systems, user authentication determines the user ID to set on outbound integrations. The receiving system extracts this information and uses the user ID to validate against the permission list to which a service operation is assigned. If the user ID is assigned to the permission list, the sender can invoke the service operation.

When using Integration Broker for integrations with other PeopleSoft systems, you do not need to set up the remote/target node as a trusted node or implement single signon for user authentication to be validated. Instead you can simply define the source system user ID(s) on the target system. The user IDs from the source system can be provisioned on the target system by Oracle Identity Manager (OIM) or another third-party provisioning application.

Note. User authentication can be implemented on PeopleTools 8.48 and later systems only.

User IDs

The PeopleSoft system can use the following methods to set the user ID in an outbound transaction:

Authentication Token When the node is a PeopleSoft (PIA) node type, the PeopleSoft system automatically generates an authentication token and includes the token in the outbound transaction.

The authentication token sets the user ID in the outbound transaction to the user ID that created the service operation.

Default User ID	The Node Definition page contains a *Default User ID* field. This is the user ID to which the node defaults, when no other user ID described in this section is set.
External Name/External Password	You can programmatically set an external name and external password in the outbound SOAP message header or query string.
External User ID/Password	The Node Definitions page contains an *External User ID* and an *External Password* field. These fields are used in conjunction with WS-Security and are used for user authentication and to set the UsernameToken credentials for WS-Security processing. The *External Password*value is optional.

On inbound integrations from a PeopleSoft node, the PeopleSoft system looks for a user ID to associate with the permission list set for a service operation in the following order.

1. Authentication token.
2. Default User ID.

On inbound integrations not from a PeopleSoft node (External nodes and third-party systems), the PeopleSoft system looks for a user ID to associate with the permission list set for a service operation in the following order.

1. External Name/External Password.
2. External User ID/External Password.
3. Default User ID.

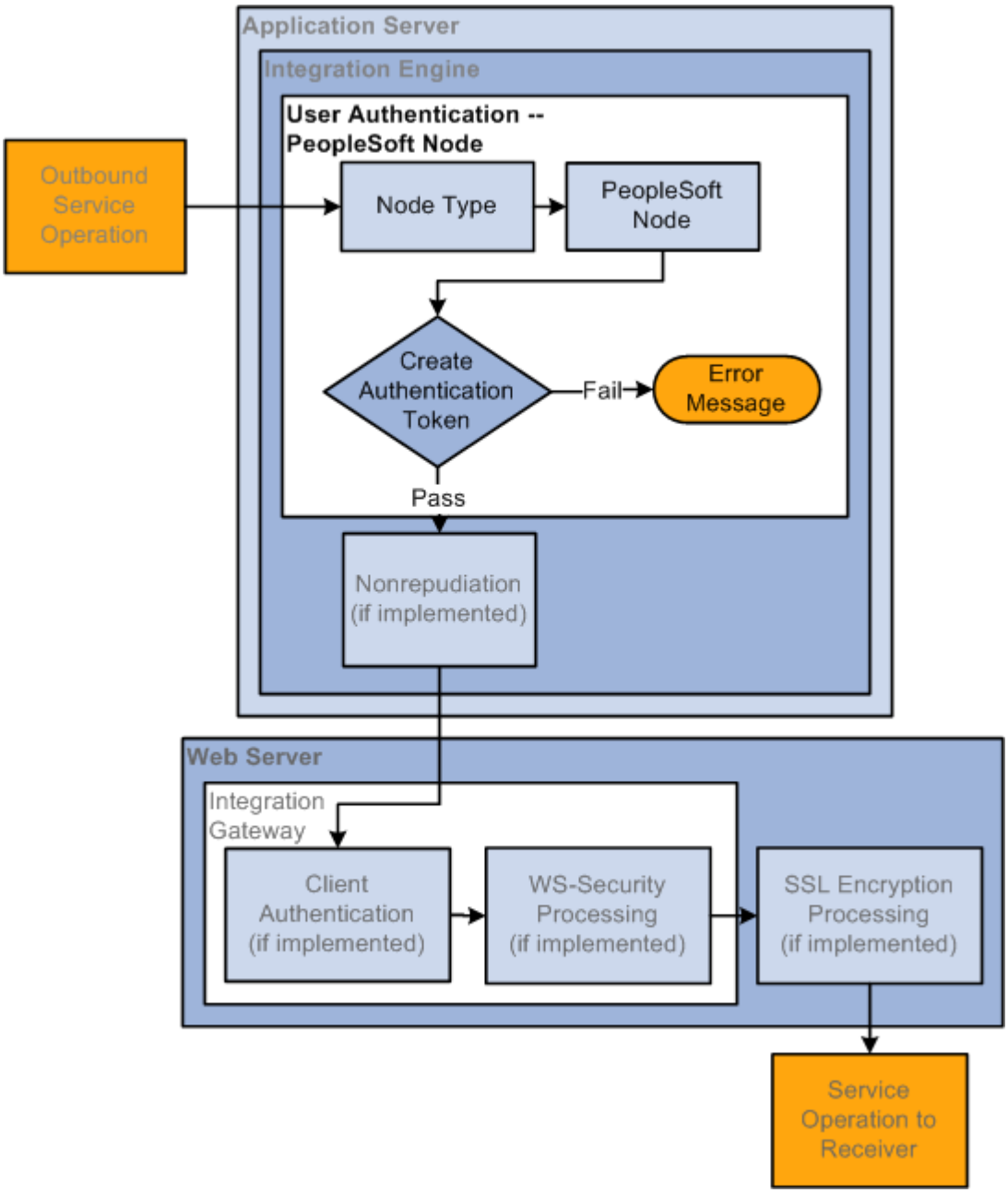
Understanding Outbound User Authentication

The outbound user authentication process determines the user ID to identify and attach to the outbound service operation. If the receiving system is a PeopleSoft system, the system validates the user ID and if the user ID belongs to the permission list to which the service operation is assigned, the service operation can be invoked.

The PeopleSoft system sets the user ID based on whether the sending node type is a PeopleSoft node (PIA) and by user ID information that may be defined in the SOAP message included with the service operation.

Outbound User Authentication: Sending Node is PeopleSoft Node Type

The following diagram illustrates the user authentication process when the local sending node is a PeopleSoft node:



Outbound User Authentication Processing when the Sending Node is a PeopleSoft Node

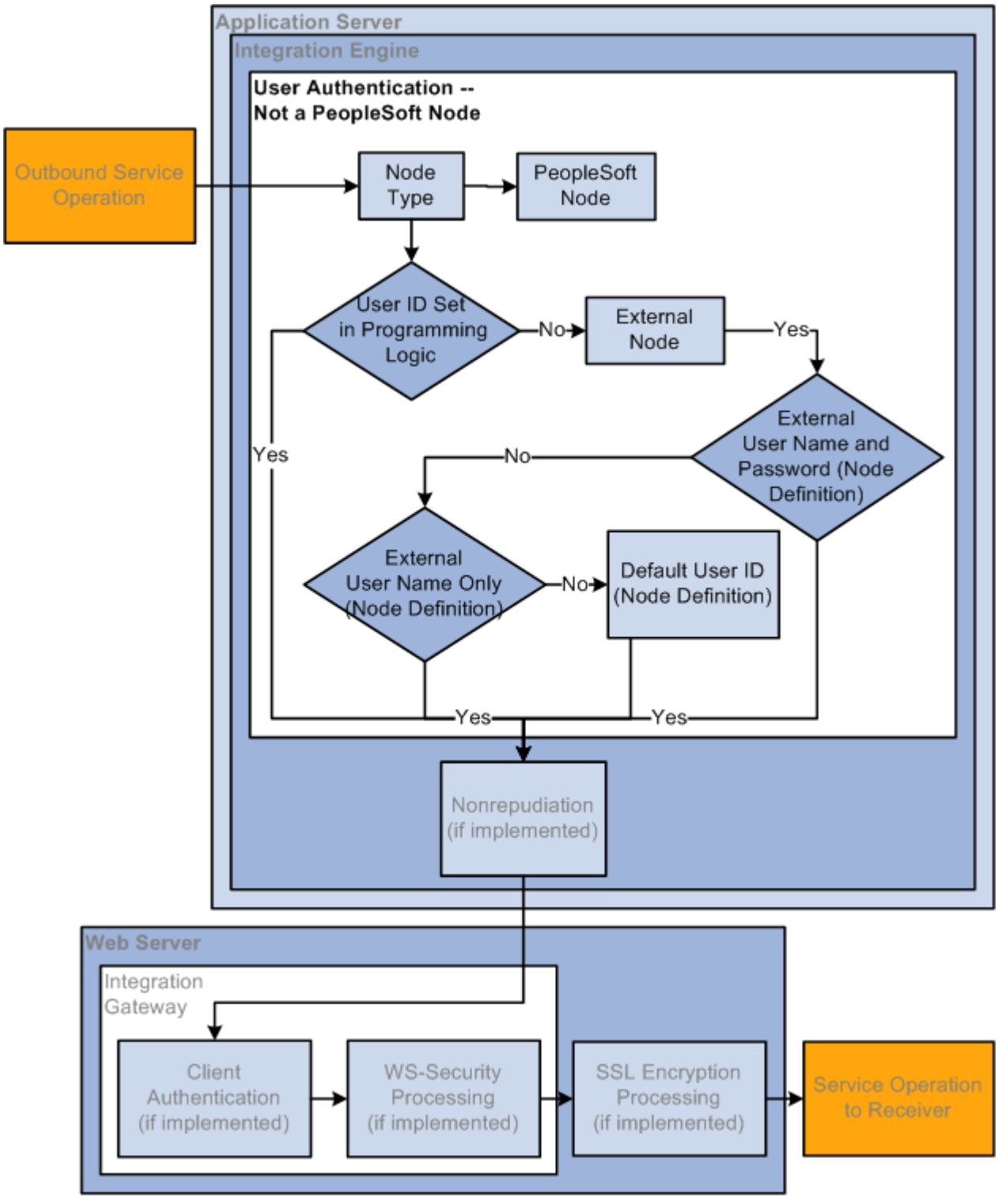
When the sending node is a PeopleSoft node, the user authentication process creates an authentication token to include in the transaction. The token is used on the receiving system to identify the sending node.

Note that the sending node does not need to be defined as trusted node on the receiving system for the PeopleSoft authentication token to be validated.

See Chapter 11, "Setting Up Secure Integration Environments," Understanding User Authentication, page 222 .

Outbound User Authentication: Sending Node is not PeopleSoft Node Type

The following diagram illustrates the user authentication process when the local sending node is not a PeopleSoft node type:



Outbound User Authentication Processing when the Sending Node is Not a PeopleSoft Node

When the sending node is not a PeopleSoft node, the system first looks at the SOAP message associated with the service operation to see if an external user ID or external user ID and password have been provided programmatically in the outbound SOAP message header. If so, the system uses that user ID/password and the service operation passes user authentication.

If an external user ID or external user ID and password are not specified programmatically in the SOAP message header, the system looks on the external node definition for user ID and password information. The system first looks for user ID and password information in the External User ID and External Password fields on the Node Definition page. If no External User ID or no External User ID/External Password is set, the system uses the Default User ID set on the Node Definitions page.

To summarize, when the sending node is not a PeopleSoft node type, the system follows this precedence for setting the user ID in the outbound service operation:

- User ID/password set in SOAP message header.
- User ID and password set in External User ID and External Password fields on the local external node definition.
- User ID set in the External User ID field on the local external node definition.
- User ID set in the Default User ID field on the local external node definition.

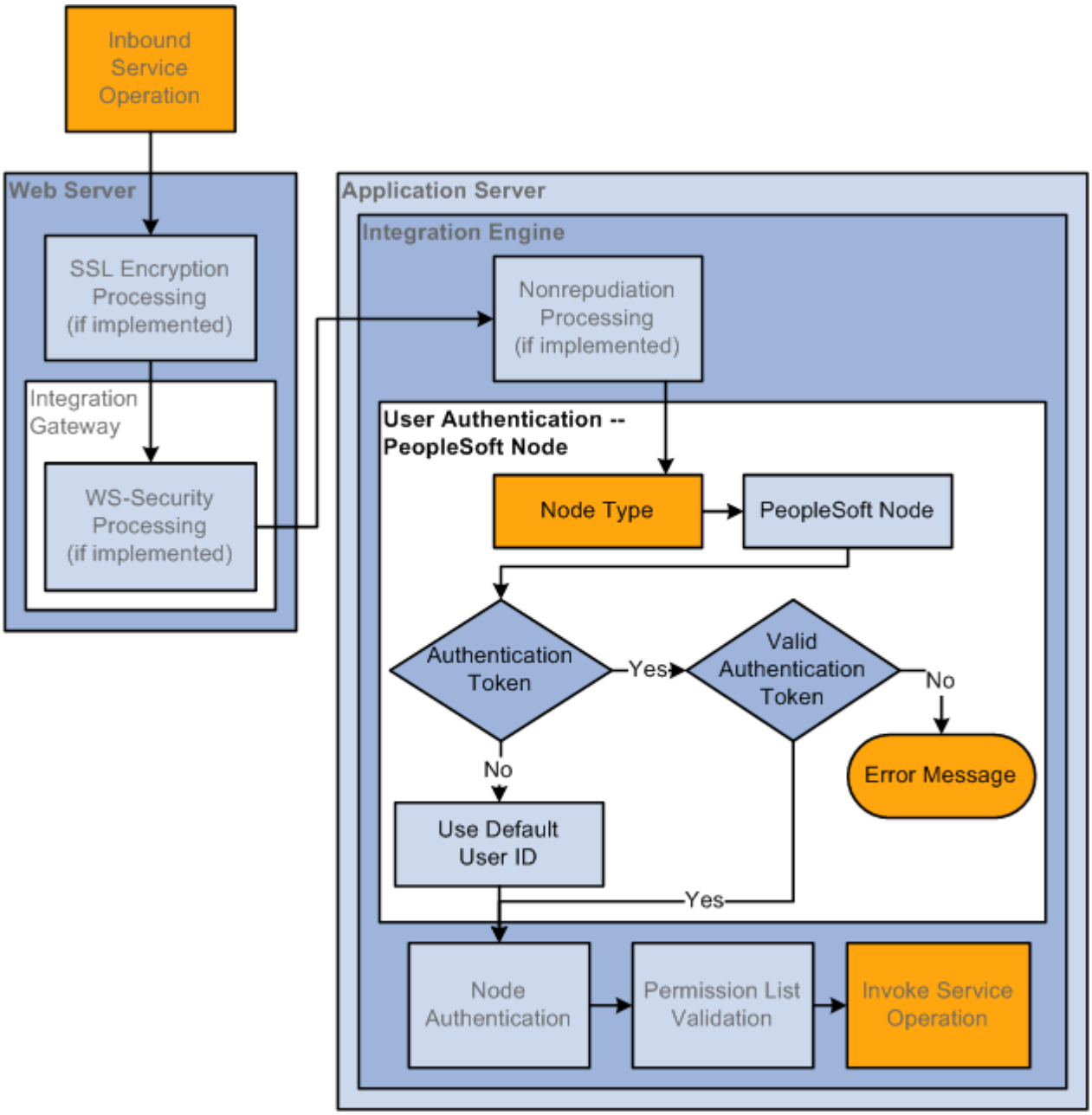
Understanding Inbound User Authentication

The inbound user authentication process determines the user ID that has been sent with an inbound service operation and determines if the sender is able to invoke the service operation.

The inbound user authentication process depends on whether the sender is a PeopleSoft node, the sender is an external node, or if the sender is not associated with any node. This section discuss user authentication processing for each of these situations.

Inbound User Authentication: PeopleSoft Node is the Sending Node

The following diagram illustrates the inbound user authentication process when a PeopleSoft node type is the sending node:



Inbound User Authentication Processing when the Sending Node is a PeopleSoft Node

If the sending node is a PeopleSoft node, the system determines if an authentication token has been sent with the transaction. The system uses the authentication token to verify the sending node.

Note that the sending node does not need to be defined as trusted node on the receiving system for the PeopleSoft authentication token to be validated.

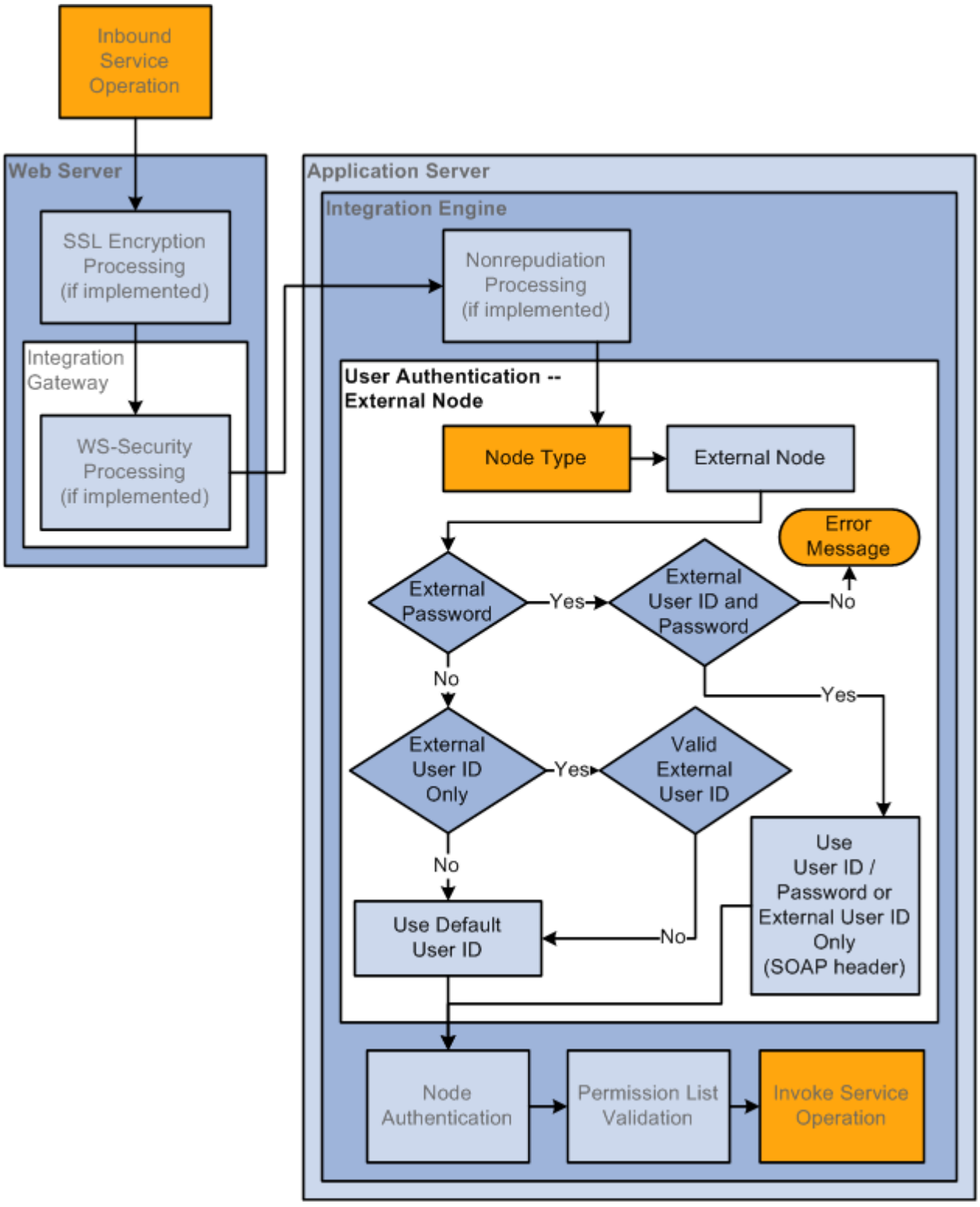
See Chapter 11, "Setting Up Secure Integration Environments," Understanding User Authentication, page 222 .

If authentication passes, the service operation has passed user authentication. If the authentication cannot be validated an error message is generated.

If no authentication token is included with the service operation, the system uses the default user ID on the external PeopleSoft node as the user ID.

Inbound User Authentication: External Node is the Sending Node

The following diagram illustrates user authentication processing when the sending node is an external node:

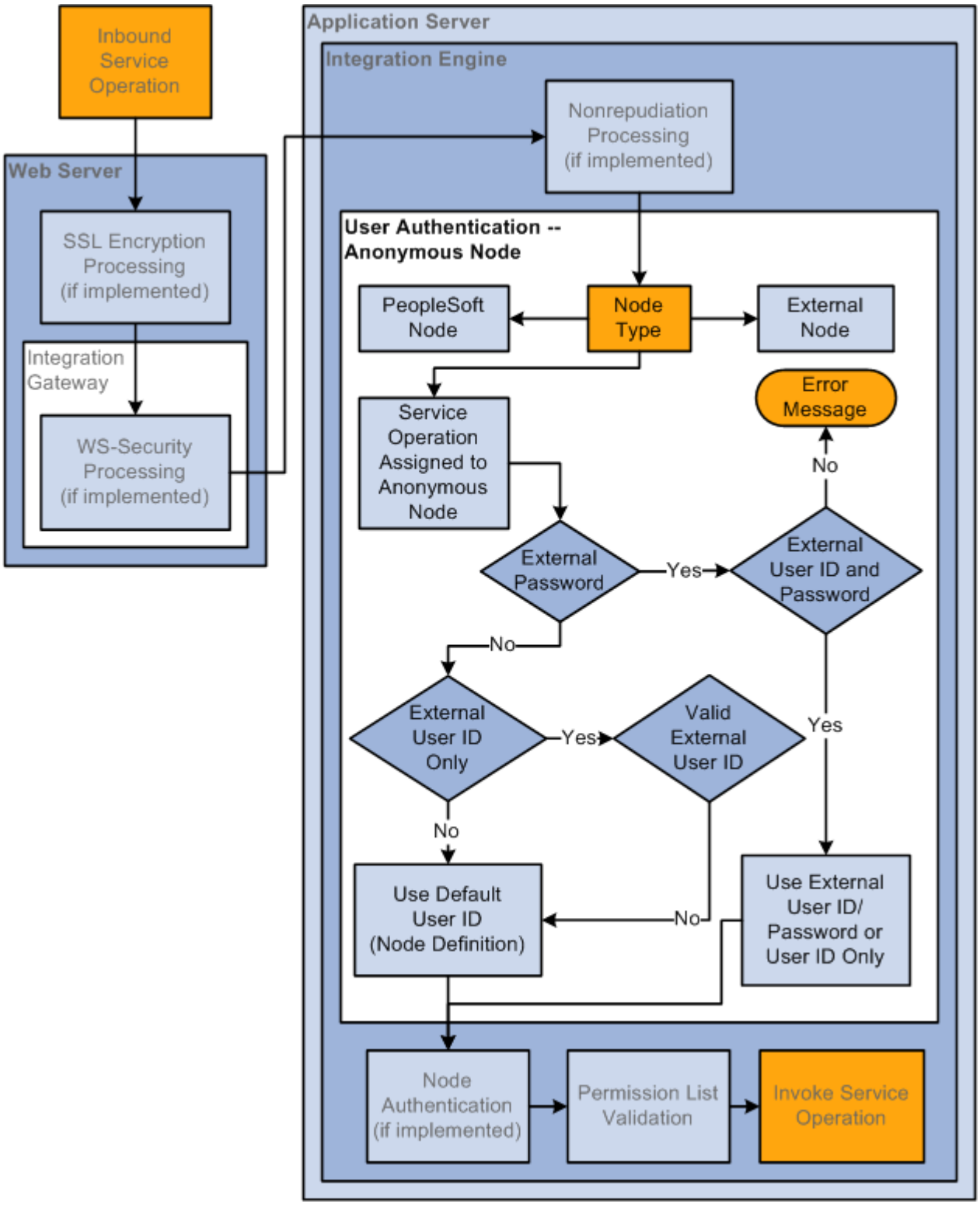


Inbound User Authentication Processing when the Sending Node is an External Node

If the sending node is an external node type, the system first looks for a user ID and password set in the SOAP message header included with the inbound service operation. If both a user ID and password are not found, the system looks in the SOAP message header for a user ID only. If no user ID/password or no user ID are found in the SOAP message header, the system uses the user ID set in the *Default User ID* field in the remote node definition.

Inbound User Authentication: Third–Party System Sending the Service Operation

The following diagram illustrates user authentication processing when a third-party system sends a service operation:



Inbound User Authentication Processing when the Sending Node is a Third-Party System

Because third-party systems do not understand the concept of a node as defined and used within the context of PeopleSoft systems, PeopleSoft assigns transactions that have no node specified to a PeopleSoft-delivered Anonymous node.

If the PeopleSoft system first checks the SOAP message header for an external name and password set programmatically.

If none is found or if the system cannot validate the user ID or password that was set programmatically, it uses the *Default User ID* set on the Node Definitions page on the remote Anonymous node definition.

Activating User Authentication on Service Operations

To activate user authentication on a service operation:

1. Access the Service Operations-General page (PeopleTools, Integration Broker, Integration Setup, Service Operations. Click the General tab.
2. Check the User/Password Required check box.
3. Save the changes.

Setting Up User Authentication on Sending Systems

This section discusses how to:

- Set up user authentication on remote PeopleSoft nodes.
- Set up user authentication on remote External nodes.
- Set up user authentication for third-party systems.

Understanding Setting Up User Authentication on Sending Systems

To set up user authentication on a sending system you must define the user ID on the remote node for the outbound transaction.

Setting Up User Authentication on Remote PeopleSoft Nodes

No set up is required to set up user authentication on a remote PeopleSoft (PIA) node type. An authentication token is automatically included in the outbound transaction. If the receiving system fails to authenticate the token an error message is returned. .

Setting Up User Authentication on Remote External Nodes

You can set the user ID for user authentication in any of the following ways on an external node:

- *External Name/Password.* Set programmatically in the SOAP message header or query string.
- *External User ID* and *External Password.* Set using the Node Definitions page.
- *Default User ID.* Set on the Node Definitions page.

Note. The user ID you specify must have access to the permission list to which a service operation is assigned to invoke the operation on the receiving system.

To access the Node Definitions page select PeopleTools, Integration Broker, Integration Setup, Nodes.

Setting Up User Authentication for Third-Party Systems

As discussed previously in this section, all inbound transactions that do not have PeopleSoft (PIA) node or external (External) node type specified are assigned to an Anonymous node.

You can set the user ID in requests from third-party systems programmatically in the external name/password elements in the outbound SOAP message header.

If the system does not find an external name or password in these locations, it uses the Default User ID field that you define on the remote Anonymous node.

See Also

Chapter 7, "Adding and Configuring Nodes," Defining Node Parameters, page 130

Excluding PeopleSoft Authentication Tokens in Outbound Requests to PeopleSoft Nodes

This section discuss how to exclude PeopleSoft authentication tokens in outbound requests to PeopleSoft nodes.

Understanding Excluding PeopleSoft Authentication Tokens in Outbound Requests to PeopleSoft Nodes

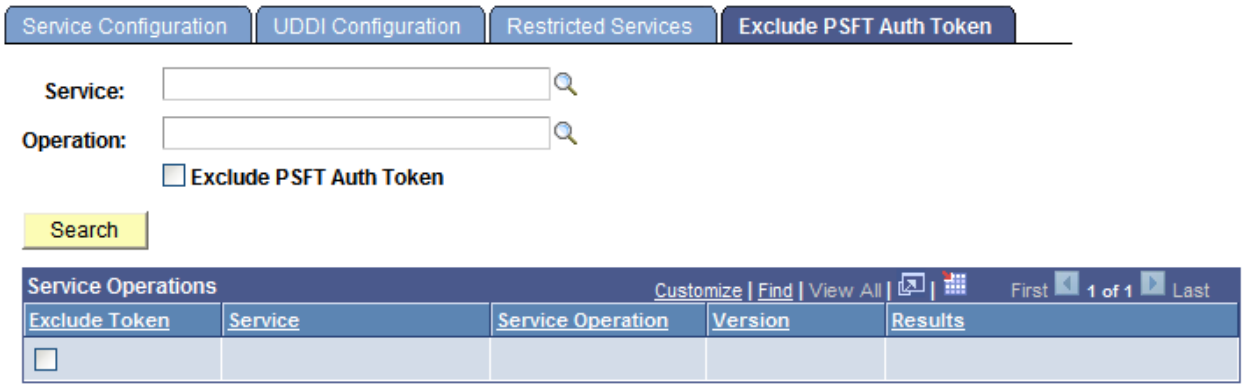
A PeopleSoft authentication token in an outbound request to a PeopleSoft target node signifies to the target PeopleSoft target system that the sender is a valid user on its system.

However, for some integrations there can be many users or validating users may not be warranted. In such cases you can exclude the PeopleSoft authentication token from inclusion in outbound requests to PeopleSoft target nodes.

When the PeopleSoft authentication token is excluded in a request, the default user ID for the sending node on the target system is the user ID used for integration authentication.

Viewing Service Operations where PeopleSoft Authentication Tokens Have Been Excluded

Use the Exclude PSFT Auth Token page (IB_SVCSETUP5) shown in the following example to view service operations where PeopleSoft authentication tokens have been excluded:



Exclude PSFT Auth Token page.

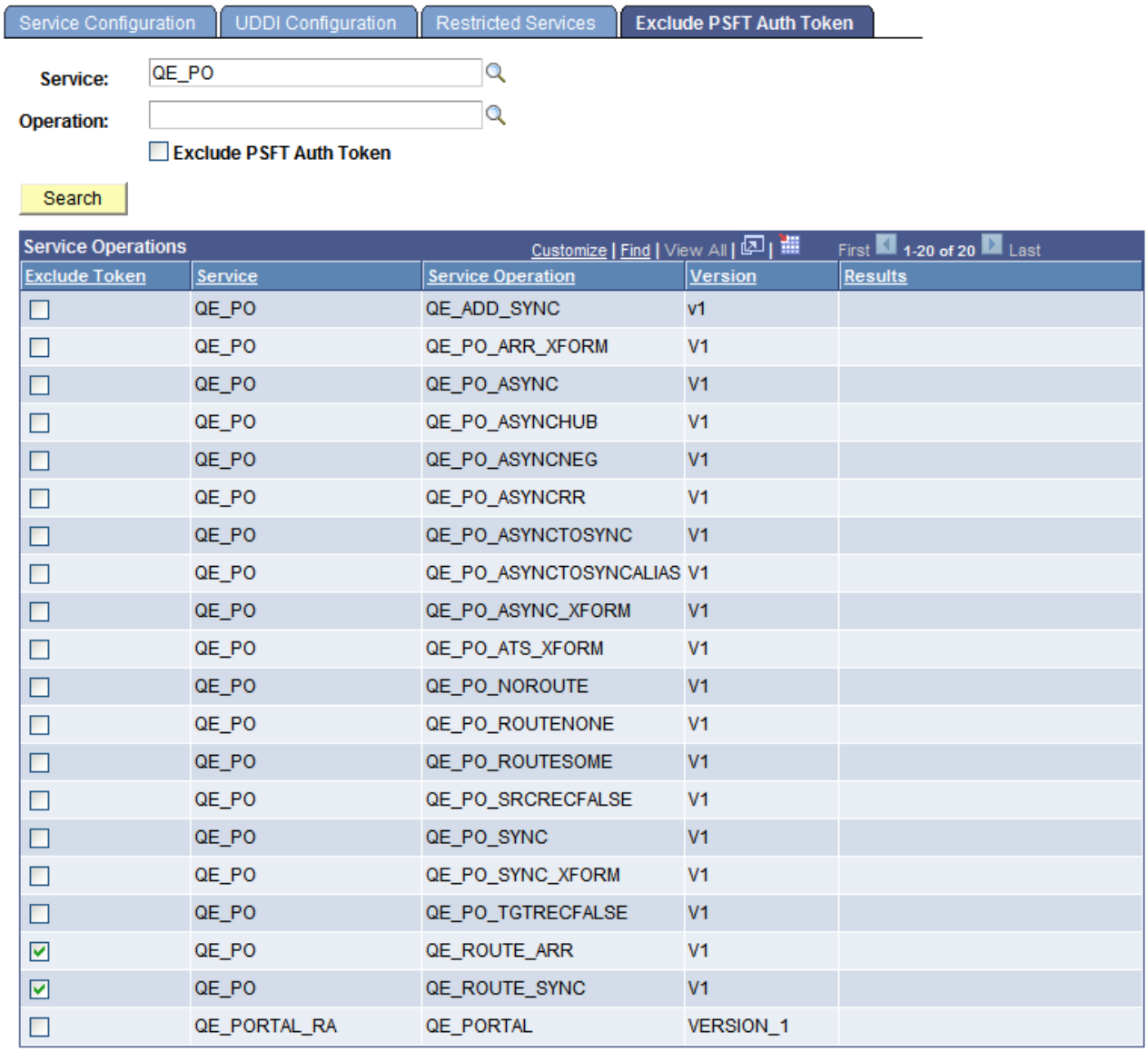
To view service operation where PeopleSoft authentication tokens have been excluded:

1. Access the Exclude PSFT Auth Token page (PeopleTools, Integration Broker, Configuration, Service Configuration, Exclude PSFT Auth Token).
2. Select the Exclude PSFT Auth Token box under the Operation field.
3. Click the Search button.

The system displays all service operations where the PeopleSoft authentication token has been excluded and will not be included in the service operation transaction.

Excluding PeopleSoft Authentication Tokens in Outbound Requests

Use the Exclude PSFT Auth Token page (IB_SVCSETUP5) shown in the following example to exclude authentication tokens in outbound requests:



Excluding the PeopleSoft authentication token from the QE_ROUTE_ARR and QE_ROUTE_SYNC service operations

In the example shown, a search was performed on the service QE_PO. The QE_ROUTE_ARR and QE_ROUTE_SYNC service operations have been selected, and therefore the PeopleSoft authentication token will be excluded from those service operations. Scrolling to the right would reveal a Results column that indicates the selection was successful.

To exclude a PeopleSoft authentication token in an outbound request:

1. Access the Exclude PSFT Auth Token page (PeopleTools, Integration Broker, Configuration, Service Configuration, Exclude PSFT Auth Token).

2. Select one or more service operations from which to exclude the PeopleSoft authentication token:

    - To select one service operation, click the Service and Operation lookup buttons to locate the service operation. Click the Exclude PSFT Auth Token box.

    - To select multiple service operations, enter all or part of the service name or service operation name. Click the Search button. A list of results displays in the Service Operations section. Check the Exclude Token box next to each service operation that should not include a PeopleSoft authentication token.

      Note that you can also click the Search button to display all service operations in the database.

3. Click the Save button.

Implementing Node Authentication

This section discusses how to:

- Set up password-based node authentication.
- Set up certificate-based node authentication.

Understanding Node Authentication

You can implement node authentication with a password or digital certificates.

Setting Up Password-Based Node Authentication

To implement password authentication, you select the *Password* option from the Authentication drop-down list in a node definition, and enter a password. When you do this for a default local node definition, you must enter the same password in any remote node definition that represents the same node on the other participating systems.

See Chapter 7, "Adding and Configuring Nodes," Defining Node Parameters, page 130.

Setting Up Certificate-Based Node Authentication

Certificate-based node authentication involves the following tasks:

- You must supply the digital certificates containing the private and public keys required for authenticated transactions.

  These elements are required at every node that participates in an authenticated transaction; PeopleSoft Integration Broker handles the mechanics of applying the keys.

  See Chapter 11, "Setting Up Secure Integration Environments," Installing Application Server-Based Digital Certificates, page 163.

- You must select the *Certificate* option from the Authentication drop-down list in a node definition.

  When you do this in a default local node definition, you must select the same option in any remote node definition that represents the same node on the other participating systems.

  See Chapter 7, "Adding and Configuring Nodes," Defining Node Parameters, page 130.

Securing Service Operations with Permission Lists

Securing service operations with permission lists is discussed elsewhere in PeopleBooks.

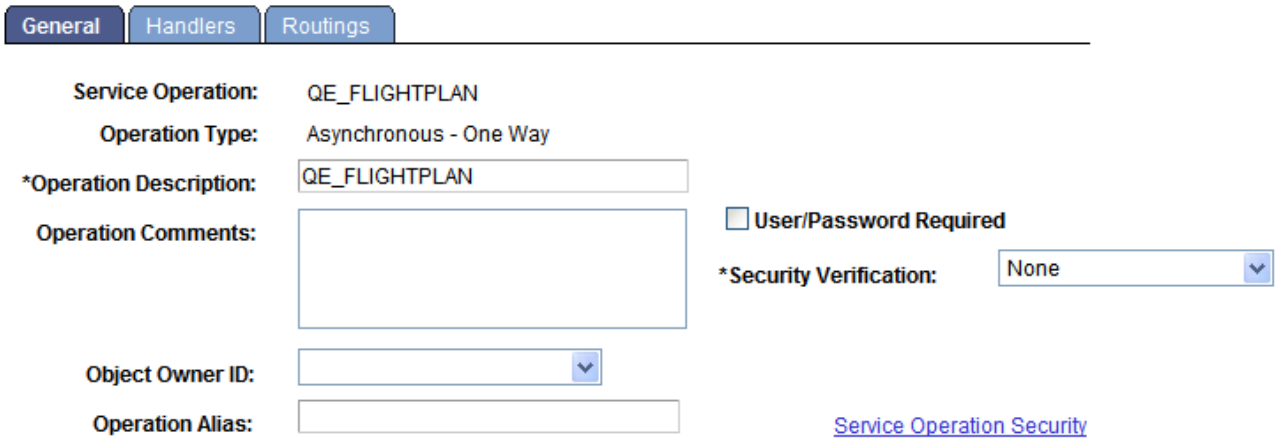
See *Enterprise PeopleTools 8.50 PeopleBook: PeopleSoft Integration Broker*, "Managing Service Operations," Setting Permissions to Service Operations.

Validating Security on Inbound Integrations

PeopleSoft Integration Broker can validate that inbound service operations from integration partners are transmitted with a level of security as determined by your organization. If they do not pass validation based on the parameters you set, the integrations are rejected.

The Service Operation page (IB_SERVICE) features a Security Verification drop-down list box that enables you to set the required level of security on inbound integrations.

The following example shows the portion of the Service Operations page that contains the Security Verification drop-down list box:



The Security Verification drop-down list box is located under the User/Password Required box.

The valid options are:

- *Digital Sign or SSL.* The integration partner must digitally sign the service operation or transmit it using SLL encryption.
- *Digitally Sign.* The integration partner must digitally sign the service operation.

- *Encrypt.* The integration partner must employ WS-Security encryption on the service operation.
- *Encrypt and Digitally Sign.* The integration partner must employ WS-Security encryption and digitally sign the service operation.
- *Encrypt or SSL.* The integration partner must employ WS-Security encryption on the service operation the service operation or transmit it using SLL encryption.
- *Encrypt/Digitally Sign or SSL.* The integration partner must employ WS-Security encryption and digitally sign the service operation, or transmit it using SLL encryption.
- *None.* (Default.) The integration partner is not required to set any specific security options on the service operation
- *SSL.* The integration partner must transmit the service operation using SLL encryption.

See Also

Chapter 11, "Setting Up Secure Integration Environments," Installing Web Server-Based Digital Certificates, page 178

Chapter 11, "Setting Up Secure Integration Environments," Implementing Web Services Security, page 192

Chapter 12

Tuning Messaging System Performance

This chapter discusses how to:

- Throttle dispatched messages through the messaging system.
- Use multi-threading to send groups of service operations in parallel.
- Implement exception handling for synchronous message processing.
- Implement master-slave dispatchers.
- Set up domain failover
- Configure integration gateways for load balancing.
- Implement load balancing on service operation queues.
- Resubmit failed transactions.
- Use the bulk load handler for large message subscriptions.
- Manage pub/sub process handler performance.

Understanding Tuning Messaging System Performance

This chapter discusses actions you can take to tune messaging system performance.

In addition, you can view messaging system performance statistics using the Service Operations Monitor.

See Also

Enterprise PeopleTools 8.50 PeopleBook: Integration Broker Service Operations Monitor, "Viewing System Performance Statistics"

Throttling Dispatched Messages Through the Messaging System

You can throttle the number of dispatched messages from a given dispatcher to its associated handler(s).

Throttling the messages that pass through the messaging system enables you to avoid Tuxedo queue saturation due to redundant Tuxedo calls, which result in degraded performance.

You can throttle messages on any of the three pub/sub dispatchers:

- PSBRKDSP
- PSPUBDSP
- PSSUBDSP

To set up dispatcher throttling, you must set the following parameters located in PSAdmin:

- Tuxedo Queue Status Check Count
- Dispatcher List Multiplier
- Dispatcher Queue Max Queue Size

Information for setting these parameters is described earlier in this PeopleBook.

See Chapter 4, "Administering Messaging Servers for Asynchronous Messaging," Specifying Dispatcher Parameters, page 23.

Using Multi-Threading to Send Groups of Messages in Parallel

This section provides an overview of multi-threading and discusses how to:

- Specify the number of available threads.
- Implement multi-threading.

Understanding Multi-Threading

Multi-threading allows you to send a group of synchronous requests in parallel, thereby eliminating the need to wait for a response for one synchronous message to be returned before you send the next synchronous message. You can also use multi-threading to send a configurable number of asynchronous message publications in parallel.

Multi-threading enables you to pool request messages into an array and make a threaded call.

When working with synchronous messages, responses are returned in an array, and are pooled in the same order in which you send them.

Multi-threading supports sender-specified routing, thereby enabling you to pass in an array of nodes on the SyncRequest call.

See Also

Chapter 12, "Tuning Messaging System Performance," Implementing Exception Handling for Synchronous Message Processing, page 242

Specifying the Number of Available Threads

The number of threads available determines the number of message you can send in parallel. For example, if there are 10 threads available, you can send 10 messages in parallel.

To specify the number of threads available for multi-threading set the Thread Pool Size parameter in PSAdmin.

The thread pool size only affects the number of messages processed at the same time, and does not limit the number of messages you can send in one API call.

Setting the Thread Pool Size for Synchronous Messaging

For synchronous messaging, set the Thread Pool Size parameter in the General Settings for Integration Broker section in PSAdmin.

For synchronous messaging, The default value is *5*. The minimum value is *1* and the maximum value is *20*.

Setting the Thread Pool Size for Asynchronous Messaging

For asynchronous messaging, set the Thread Pool Size parameter in the Settings for Publication Contract Handler section in PSAdmin.

For asynchronous messaging, The default value is *1*. The minimum value is *1* and the maximum value is *20*.

Implementing Multi-Threading

This section provides the syntax for multi-threading and provides a synchronous multi-threading code example.

Syntax

The syntax for implementing multi-threading is:

```
Array of messages = %IntBroker.SyncRequest(Array of messages, array of
sender-specified routing);
```

The IntBroker object is responsible for managing the messages, instantiation of the SyncRequest handler and calling the Send method for each request. The IntBroker object then polls the SyncRequest handler object to determine when all processing is complete. At that time, status and error checking is performed and the response message objects are created. The response messages are packaged as an array and returned to the calling method.

Synchronous Multi-Threading Example

The following example shows code for synchronous multi-threading


```
Local Rowset &FLIGHTPLAN, &FLIGHTPLAN_RETURN;
Local Message &MSG;

Local array of Message &messages;
Local array of Message &return_mesages;

&messages = CreateArrayRept(&MSG, 2);
&return_mesages = CreateArrayRept(&MSG, 2);

&FLIGHT_PROFILE = GetLevel0();
&messages [1] = CreateMessage(Message.QE_FLIGHTPLAN_SYNC);
// populate the rowset
&messages [1].CopyRowset(&FLIGHT_PROFILE);

&messages [2] = CreateMessage(Message.QE_FLIGHTPLAN_SYNC);
// populate the rowset
&messages [2].CopyRowsetDelta(&FLIGHT_PROFILE);

&return_mesages = %IntBroker.SyncRequest(&messages);

// process the return rowset
&FLIGHTPLAN_RETURN = &return_mesages [1].GetRowset();
&temp = &return_mesages [1].GenXMLString();

// process the return rowset
&FLIGHTPLAN_RETURN = &return_mesages [2].GetRowset();
&temp = &return_mesages [2].GenXMLString();
```

See Also

Enterprise PeopleTools 8.50 PeopleBook: PeopleCode API Reference, "Message Classes," SyncRequest

Implementing Exception Handling for Synchronous Message Processing

When a an outbound synchronous request fails you can throw a framework exception leading to a message box error and subsequent component roll back of the transaction.

Note. This type of exception handling applies to outbound synchronous requests only, including outbound multi-threaded synchronous requests.

For example, if 10 synchronous requests are performed in parallel (threaded sync request), you have the option to select the User Exception check box on the routing definition for the service operation. When the User Exception check box is selected, if any of the synchronous requests error, the component is not rolled back. You can check each synchronous request to determine if there is an error and actually read the associated error message. You can then throw an exception or go on to process the next synchronous request in the array.

See *Enterprise PeopleTools 8.50 PeopleBook: PeopleSoft Integration Broker*, "Managing Service Operation Routing Definitions."

The following example shows sample pseudo PeopleCode to read the exception:


```
Local Rowset &FLIGHTPLAN, &FLIGHTPLAN_RETURN;
Local array of Message &messages;
Local array of Message &return_mesages;

&messages = CreateArrayRept(&MSG, 2);
&return_mesages = CreateArrayRept(&MSG, 2);

QE_FLIGHTDATA.QE_ACNUMBER.Value = QE_FLIGHTDATA.QE_ACNUMBER + 1;

&FLIGHT_PROFILE = GetLevel0();

&rs1 = &FLIGHT_PROFILE.GetRow(1).GetRowset(Scroll.QE_NAVIGATION);
&rs2 = &FLIGHT_PROFILE.GetRow(1).GetRowset(Scroll.QE_RADAR_PRESET);
&rs3 = &FLIGHT_PROFILE.GetRow(1).GetRowset(Scroll.QE_ARMAMENT);
&messages [1] = CreateMessage(Operation.SYNC_PARTS);

For &i = 1 To &messages [1].PartCount

   If &i = 1 Then
      &rs1.CopyTO(&messages [1].GetPartRowset(&i));
   End-If;

   If &i = 2 Then
      &rs2.CopyTO(&messages [1].GetPartRowset(&i));
   End-If;

   If &i = 3 Then
      &rs3.CopyTO(&messages [1].GetPartRowset(&i));
   End-If;

End-For;

&messages [2] = CreateMessage(Operation.SYNC_PARTS);

For &i = 1 To &messages [2].PartCount

   If &i = 1 Then
      &rs1.CopyTO(&messages [2].GetPartRowset(&i));
   End-If;

   If &i = 2 Then
      &rs3.CopyTO(&messages [2].GetPartRowset(&i));
   End-If;

   If &i = 3 Then
      &rs2.CopyTO(&messages [2].GetPartRowset(&i));
   End-If;

End-For;

&return_mesages = %IntBroker.SyncRequest(&messages);

If &return_mesages [1].ResponseStatus = %IB_Status_Success Then

   For &i = 1 To &return_mesages [1].PartCount

//perform local processing on response data

   End-For;

Else

   &nMsgNumber = &return_mesages [1].IBException.MessageNumber;
   &nMsgSetNumber = &return_mesages [1].IBException.MessageSetNumber;
```

```
  &exceptString = &return_mesages [1].IBException.ToString();
// Evaluate exception and throw error if necessary

End-If;

If &return_mesages [2].ResponseStatus = %IB_Status_Success Then

   For &i = 1 To &return_mesages [2].PartCount

  //perform local processing on response data   End-For;

  Else

   &nMsgNumber = &return_mesages [2].IBException.MessageNumber;
   &nMsgSetNumber &return_mesages [2].IBException.MessageSetNumber;
  &exceptString = &return_mesages [2].IBException.ToString();

// Evaluate exception and throw error if necessary

End-If;
```

Implementing Master-Slave Dispatchers

This section provides an overview of master-slave dispatching and describes how to:

- Configure dynamic slave dispatchers.
- Configure static slave dispatchers.
- Create template slave domains.
- Implement master-slave load balancing.
- Implement deferred master domain processing.

Understanding Implementing Master-Slave Dispatchers

Master-slave dispatching is where a master domain allocates messages to one or more slave dispatchers for processing. This section provides an overview of master-slave dispatcher processing.

Master-Slave Dispatcher Processing

A slave dispatcher processes service operations assigned to it by a master dispatcher.

A master domain allocates service operations to a slave for processing when:

- The master detects that a slave dispatcher is active and not busy processing service operations.
- The slave has an active queue on which the master is currently processing service operations.

The dispatcher(s) processing in slave mode then process the allocated service operations.

Master and slave dispatchers can reside on the same or on different machines.

Note. When master-slave processing is implemented, there can be only one master domain at a given time.

You can create a domain consisting of only dedicated slave pub/sub servers. These servers register themselves as slaves, along with additional configurable information, such as the number of process handlers booted, so that the appropriate master server can use that information to allocate work (service operations to process) to the slave server(s).

The master domain can allocate work to one or more slave domains.

Slave Types

There are two types of dispatcher slaves:

Dynamic slaves

A dynamic slave can change from a master to a slave.

Dynamic slaves are configured in conjunction with domain failover. If a slave domain has the highest priority within a failover group, it can dynamically change to a master during failover.

You configure dynamic slaves in the Failover Configuration page in the Service Operations Monitor.

Static slaves

Static slaves are those that cannot become masters without manual configuration.

You configure static slave domains in PSAdmin.

Template slaves

A template slave is an already-configured master domain that you import into PSAdmin an save as a slave domain.

Template slaves enable you to dynamically add slave domains without performing any configuration changes in PSAdmin. You use the Import Domain Configuration command in PSAdmin to import a master domain configuration and then save it as a static slave domain.

The slave domain created uses all the Pub/Sub processes and queue lists configured for the master domain on which the slave template is based. If dedicated servers are configured for the master domain, they are also imported an available on the slave domain.

Failover and Master-Slave Dispatchers

You can create a slave domain for use in domain failover.

The domain with the highest priority dynamically becomes the active domain (master domain) in each group during failover. The next domain in priority will be programmatically configured as an active slave domain.

When a failover occurs the domain that failed becomes inactive. The failover domain specified goes from an active slave to an active master. The next domain in priority then becomes an active slave.

You can set failover for slave dispatchers. However, slave dispatchers cannot be part of any group and you cannot prioritize them.

See Chapter 12, "Tuning Messaging System Performance," Setting Up Domain Failover, page 255.

Configuring Dynamic Slave Dispatchers

Use the Failover Configuration page in the Service Operations Monitor to configure dynamic slave dispatchers.

See Chapter 12, "Tuning Messaging System Performance," Setting Up Domain Failover, page 255.

Configuring Static Slave Dispatchers

You use PSAdmin to specify a pub/sub dispatcher as master or slave by setting the following property:

Property	*Description*
DispatcherSlaveMode	Options are: • *0*: Master mode. (Default.) • *1*: Slave mode.

See Also

Enterprise PeopleTools 8.50 PeopleBook: System and Server Administration, "Using the PSADMIN Utility"

Creating Template Slave Domains

This section discusses how to:

- Import domain configurations from application domains.
- Import domain configurations from files.
- View template slave domains
- Add and delete dispatcher queues from template slave domains.
- Restore dispatcher queue lists.

Understanding Template Slave Domains

Template slaves enable you to dynamically add slave domains without performing any configuration changes in PSAdmin .

When you create a template slave domain you import a domain configuration and save it as a template for the slave domain. This process creates a static slave domain that uses all of the pub/sub processes and queue lists configured for the domain that you import. If dedicated servers are configured for the domain that you import, they are imported and available on the slave domain. After you import a domain configuration and save it as a slave domain, you can add or remove dispatcher queues from the slave domain as needed.

All template slave domains must be based on the same master domain configuration.

Understanding Importing Domain Configurations

To import the domain configuration on which the template slave domain is based, you use the Import Domain Configuration command in PSAdmin . You can import a domain that is already configured in PSAdmin or you can import a domain configuration from a file.

When the configuration is imported, all the Pub/Sub processes are configured identically to the domain on which the slave template is based, including the PSWATCHSRV and PSMONITOR server processes. Other process, such as PSAPPSRV, PSSAMSRV, and so on, are not included in the template slave domain, as it is intended specifically for use with PeopleSoft Integration Broker. However, you can modify the template slave domain configuration file to include these processes if needed.

To import a domain configuration from an application domain you specify the location of <PS_CFG_HOME> for the domain that you want to import. For example, the location might be *c:\documents and settings\admin\psft\ps\<PS_CFG_HOME>*.

To import a PeopleTools 8.49 or earlier application domain you must specify the <PS_HOME> location for <PS_CFG_HOME>. For example, the location might be *c:\documents and settings\<PS_HOME>\<PS_CFG_HOME>*.

Prerequisites for Importing Domain Configurations

If you are importing a domain that is already configured in PSAdmin, you must first set the PS_FILEDIR environment variable equal to the PS_HOME location of the domain you importing.

If importing a domain configuration from a file, you must first set the PS_FILEDIR environment variable to the location where you are importing the file.

Importing Domain Configurations from Files

To import a domain configuration from a file:

1. Open PSAdmin.

   The PeopleSoft Server Administration menu appears.

2. Enter *1* for Application Server and press the ENTER key.

   The PeopleSoft Application Server Administration menu appears.

3. Enter *4* for Import Domain Configuration and press the ENTER key.

   The PeopleSoft Import Application Server Configuration menu appears.

4. Enter *2* for Import IB Master Configuration and press the ENTER key.

5. Enter *1* for Import from file and press theENTER key.

   A prompt displays to enter the full path to the domain configuration file to import.

6. Enter the full path to the domain configuration file to import and press the ENTER key.

   A prompt displays to enter a name for the new domain.

7. Enter a name for the new domain and press the ENTER key.

The system merges the domain configuration with the new template slave domain and creates the new configuration and loads it on the application server. Upon completion, the PeopleSoft Domain Administration menu appears, where you may boot the template slave domain, configure the template slave domain or perform other administrative tasks.

Importing Domain Configurations from Application Domains

To import a domain configuration from an application domain:

1. Open PSAdmin.

   The PeopleSoft Server Administration menu appears.

2. Enter *1* for Application Server and press the ENTER key.

   The PeopleSoft Application Server Administration menu appears.

3. Enter *4* for Import Domain Configuration and press the ENTER key.

   The PeopleSoft Import Application Server Configuration menu appears.

4. Enter *2* for Import IB Master Configuration and press the ENTER key.

5. Enter *2* to for Import from application domain and press theENTER key.

   A prompt displays to enter the location of <PS_CFG_HOME>.

6. Enter the location of <PS_CFG_HOME> and press theENTER key.

   The Tuxedo Domain List appears that lists the application domains that you can import.

7. Enter the number that corresponds to the application domain to import.

   A prompt displays to enter a name for the new domain.

8. Enter a name for the new domain and press the ENTER key.

The system merges the domain configuration with the new template slave domain and creates the new configuration and loads it on the application server. Upon completion, the PeopleSoft Domain Administration menu appears, where you may boot the template slave domain, configure the template slave domain or perform other administrative tasks.

Adding and Removing Dispatcher Queues from Template Slave Domains

Template slave domains contain all of the dispatcher queues that exist on the domain on which it is based. However, you can add and remove queues to configure the template slave domain to suit your requirements.

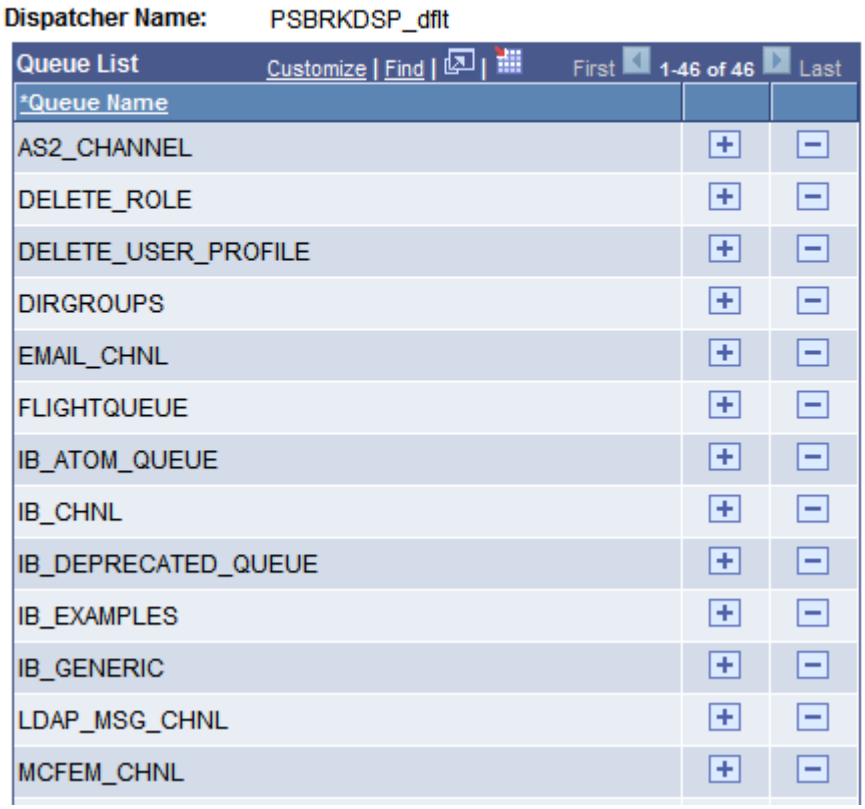
Note. Before you can add or remove queues from a template slave, you must inactivate all domains.

The Slave Templates page (IB_DOMAIN2_SEC) lists the dispatcher queues assigned to each dispatcher process of the template slave. The following example shows a partial view of the Slave Templates page:

Use the Slave Templates page to view the queues assigned to each dispatcher process of a template slave.

From the Slave Templates page, you can use the Add/Remove Queues link located under each dispatcher process name to access the Add/Remove Queues page to add or remove queues assigned to the dispatcher. The following example shows the Add/Remove Queues page:

Add/Remove Queues



Use the Add/Remove Queues page to add or remove queues for a dispatcher process of a template slave

Note that the previous graphic shows a partial queue list for the domain.

On the Add/Remove Queues page you can use the plus (+) button to add a queue to the queue list. Use the minus (-) button to remove a queue from the list.

To add or remove dispatcher queues for template slaves:

1. Access the Domain Status page (PeopleTools, Integration Broker, Service Operations Monitor, Administration, Domain Status).
2. Check the All Domains Inactive box and click the Update button.

   The Force Reset button appears.
3. Click the Force Reset button to reset any contacts that are in a *Started* or *Working* state.
4. Click the Slave Templates link.

   The Slave Templates page appears.
5. Click the Add/Remove Queues link for a queue dispatcher.

   The Add/Remove Queues page appears and displays the queue list for the dispatcher.

6. To add a queue:
   a. Click the plus (+) button to insert a new row into the list.
   b. Click the Lookup button to search for a queue to add.
   c. Click the OK button at the bottom of the Add/Remove Queues page.

      The Slave Templates page appears.
   d. Click the Update button.

      The Domain Status page appears.
7. To delete a queue:
   a. Click the minus (-) button next to the queue to delete.
   b. Click the OK button in the dialog box to confirm the delete action.
   c. Click the OK button at the bottom of the Add/Remove Queues page.

      The Slave Templates page appears.
   d. Click the Update button.

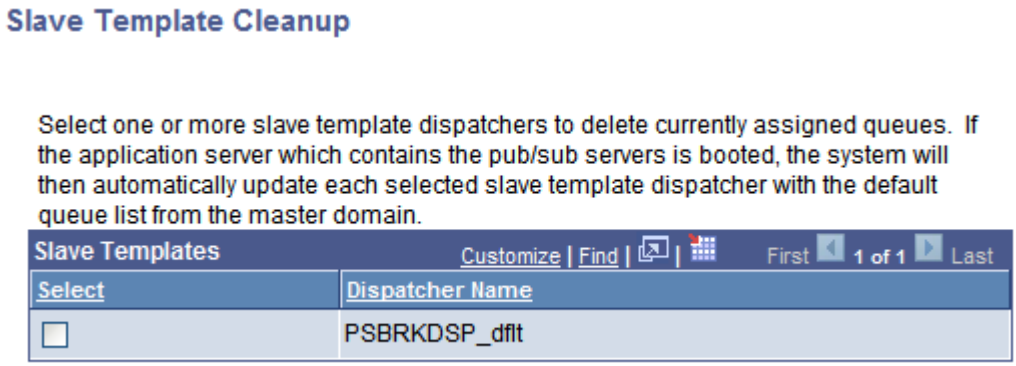
      The Domain Status page appears.
8. Activate the domains in the messaging system.

   Check the All Domains Active box and click the Update button to activate the domains.

Restoring Template Slave Dispatcher Queue Lists

Restoring a queue list deletes any changes you have made to a template slave dispatcher queue list, and restores it to the queue list that you originally imported from the master domain.

To restore a template slave queue list, use the Slave Template Cleanup page (IB_DOMAIN3_SEC) shown in the following example:



Use the Slave Template Cleanup page to restore dispatcher queue lists to the default settings from the master domain.

In the previous example, PSBRKDSP_dflt appears in the Dispatcher Name field, meaning that additions or deletions have been made to the queue. In this example, you could use the page to restore the default template slave dispatcher list for PSBRKDSP_dflt.

To restore a template slave dispatcher queue list, you must first inactivate all domains on the system.

To restore a template slave dispatcher queue list:

1. Inactivate the domains on the messaging system:
    a. Access the Domain Status page (PeopleTools, Integration Broker, Service Operations Monitor, Administration, Domain Status).
    b. Check the All Domains Inactive box and click the Update button.

       The Force Reset button appears.
    c. Click the Force Reset button to reset any contacts that are in a Started or Working state.
2. Access the Slave Template Cleanup page:
    a. From the Domain Status page, click the Slave Templates link.

       The Slave Templates page appears.
    b. On the Slave Templates page, click the Slave Template Cleanup link.

       The Slave Template Cleanup page appears.
3. Check the box next to each dispatcher process name for which you want to restore the default queue list.
4. Click the Delete button.

   The Domain Status page appears.
5. Activate the domains in the messaging system.

   Check the All Domains Active box and click the Update button to activate the domains.

Implementing Master-Slave Load Balancing

This section provides and overview of master-slave load balancing and discusses how to set up master-slave load balancing on the PeopleSoft system.

Understanding Master-Slave Load Balancing

You can implement master-slave load balancing on the integration system to compensate for processing capabilities of various machines on which master domains and slave domains run.

As an example, you might have a domain on machine that is also running the PeopleSoft Pure Internet Architecture. In this case, you could configure master-slave load balancing such that the machine that is running the PeopleSoft Pure Internet Architecture processes fewer requests than other machines on which domains reside.

Another example is a situation where the machines on which you are running domains have different processing capabilities due to the hardware installed in them. In this situation you can configure the machines with the most process power to process the greater number of requests.

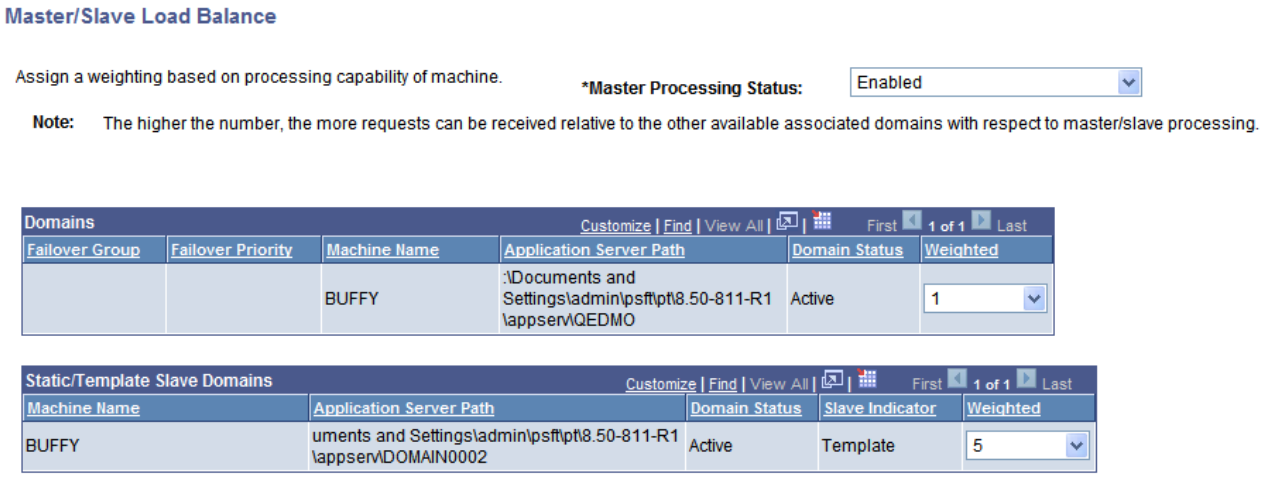
To configure master-slave load balancing, you assign a weight between*1* and *10* to each domain to distribute request processing. A domain assigned a weighted value of *1* processes the fewest requests; a domain assigned a weighted value of *10* processes the greatest number of requests.

Setting Up Master-Slave Load Balancing

You set up master-slave load balancing using the Master/Slave Load Balancing page (IB_DOMAIN_SEC) shown in the following example:

To set up master/slave load balancing, you assign a processing weight value to each domain. The domain with the lowest number processes the fewest number of requests and is the master domain. The domains with the higher numbers are the slave domains and process the greatest number of requests.

The following example shows master/slave load balancing set up for the system:



Setting up master-slave load balancing using the Master/Slave Load Balance page.

The Domain section on the page lists information for the master domain, while the Static/Template Slave Domains section lists information about static slave domains.

The example shows two domains configured on one machine. The domain listed in the Domains section, *QEDMO*, is the master domain and has a load balance weight of *1* assigned to it. Given the load balance weight assigned to the domain, it processes the fewest number of requests.

The domain listed in the Static/Template Slave Domains section, *DOMAIN002*, has a load balance weight of *5* assigned to it. It processes a greater number of requests than the master domain.

To set up master-slave load balancing:

1. Access the Master/Slave Load Balance page (select PeopleTools, Integration Broker, Service Operations Monitor, Administration, Domain Status. Click the Master/Slave Load Balance link.)

2. For each domain select a value from the Weighted drop-down list box to assign a load balancing weight for the domain.

3. Click the OK button.

Implementing Deferred Master Domain Processing

This section provides an overview of deferred master domain processing and discusses how to set up deferred master domain processing.

Understanding Deferred Master Processing

PeopleSoft Integration Broker enables you to defer request processing on master domain to slave domains that are available for processing. Configuring deferred master processing enables you to free processing resources on the master domain machine due to hardware or processing power limitations, or so it can run other processes.

The Master/Slave Load Balance page features a Master Processing Status drop-down list box where you set the processing status for the master domain. The following table lists the master processing statuses and their descriptions.

Master Domain Processing Status	*Description*
Enabled	The master domain processes its appropriate share of requests. (Default.)
Deferred – All Queues	The master domain does not send any requests to its respective process handler(s) as long as there is at least one active slave domain that can be used for the dispatch cycle.
Deferred – Unordered Queues	The master domain does not send any requests in an unordered queue to its respective process handler(s) as long as there is at least one active slave domain that can be used for the dispatch cycle. When you select this option the master domain dispatchers only send requests to a slave domain for processing if the queue being processed is defined as unordered queue. If the queue is not unordered, the master domain sends the request to its own process handler for processing not to the slave domain.

If the system is set to any of the deferred modes the master will process requests if no slave dispatchers are available. In each of the deferred modes, the master assigns processing to slave dispatchers based on the load balancing weight value assigned to the slave dispatcher.

Setting Up Deferred Master Domain Processing

To set up deferred master domain processing:

1. Select PeopleTools, Integration Broker, Service Operations Monitor, Administration, Domain Status.

   The Domain Status page appears.

2. Click the Master/Slave Load Balance link.

   The Master/Slave Load Balance page appears.

3. From the Master Processing Status drop-down list box, select a master domain processing status.

   The valid values are:

   - *Enabled*
   - *Deferred — All Queues*
   - *Deferred — Unordered Queues*

4. Click the OK button.

Setting Up Domain Failover

This section discusses how to:

- Enable failover on domains.
- Set up dynamic master-slave dispatchers.
- Check the validity of queue sets.
- View queues assigned to failover groups.

Understanding Domain Failover

This section discusses domain failover.

Domain Failover

Domain failover ensures that PeopleSoft Integration Broker continues processing message requests and responses, even if it incurs errors or other problems on the primary domain. When failover is activated, service operation processing will switch to back up domains should Integration Broker incur any errors or problems on the primary domain. In addition, should the domain fail, you can send a system-generated email notification to individuals.

If the connection with the database is lost and the handlers are processing service operations at that time, the handlers attempt to reboot. If initialization fails, the handlers are not rebooted. If you are using failover, the failover process takes over and failover of the domain occurs.

Domain Failover Groups

You can set up domain failover groups, so that all failover takes place on specific domains. To set up failover groups, you assign a domain a failover group number. After you assign domains to a group, you then assign the failover priority for all domains within the group.

Note. If you do not use dedicated messaging servers, you typically do not need to use failover groups.

Note. Queue sets within failover groups must be identical; queue sets between failover groups must be unique.

The example of the Failover Configuration page in the Enabling Failover on Domains section in this chapter shows five domains attached to the application server. The first three domains have been assigned to failover group one, as indicated by the value *1* in the Failover Group field for each domain. The last two domains have been assigned to failover group two, as indicated by the value *2* in the *Failover Group* field for each domain.

The failover priority within group one is as follows: the first domain in the list, *QEDMO,* is the main and primary back-up domain as indicated by the failover priority value *1*; the second domain in the list, *DOMAIN02,* is the second back-up domain as indicated by the failover priority value *2*; the third and final back-up domain for failover group one is indicated by the failover priority value *3* and is the domain *DOMAIN03.*

The failover priority within group two is as follows: the fourth domain in the list, *DOMAIN04,* is the first back-up domain for group two as indicated by the failover priority value *1*; the last domain in the list, *DOMAIN05,* is the second back-up domain as indicated by the failover priority value *2.*

Failover Groups — Priority Reset

PeopleSoft Integration Broker features a priority reset option that works in conjunction with failover groups.

When you set this option and failover occurs, the system attempts to use the domain defined as group priority value of 1 before failing over to the next sequential domain.

As an example, you could have a failover group defined as follows:

Group Failover Priority	*Domain Name*
1	QEDMO
2	DOMAIN02
3	DOMAIN03

In our example, domain *QEDMO* has already failed over to domain *DOMAIN02.* If domain *DOMAIN02* subsequently fails, the system attempts to failover back to the domain with the highest failover priority setting, domain *QEDMO.* If the system is unsuccessful in failing over to domain*QEDMO,* domain *DOMAIN02* fails over to the next sequential domain in the failover group, domain *DOMAIN03.*

You set the Priority Reset option on the Failover Configuration page.

See Chapter 12, "Tuning Messaging System Performance," Enabling Failover on Domains, page 257.

Dynamic and Static Master-Slave Dispatchers

You can implement master-slave dispatchers in conjunction with domain failover.

When *dynamic* slaves are implemented, the domain with the highest priority will become the active domain (master domain) in each group during failover. The next domain in priority is automatically programmatically configured as an active slave domain. You configure dynamic slaves in the Service Operations Monitor.

Static domain slaves are always slaves. You configure static slaves in PSAdmin.

Failover Priority — General Failover

In general domain failover, if a failover domain becomes inactive, the system attempts failover back to the highest priority domain. If it is unable to do so, the next domain prioritized becomes the active domain.

As an example, consider an integration system with the domains and failover priorities shown in the following table:

Domain	Failover Priority
Domain A	1
Domain B	2
Domain C	3

In this integration system if Domain A fails, the system will failover to Domain B. If Domain B later fails, the system attempts to failover back to Domain A. If it is unable to do so, the system will failover to Domain C.

Failover Priority — Dynamic Slave Failover

In dynamic slave failover, if the dynamic slave fails, the system generates and email notification and the dynamic slave becomes inactive. The system does not failover to find another dynamic slave.

Failover on the primary system has to occur for a dynamic slave to automatically go into effect.

Failover Priority Modification

If you modify failover priorities when failover is enabled and change the priorities of the current active domain, all domains are reset to inactive and the domain with the priority value of *1* is activated. However, if failover is not active and you change priorities, PeopleSoft Integration Broker saves the changes without any domain status reset.

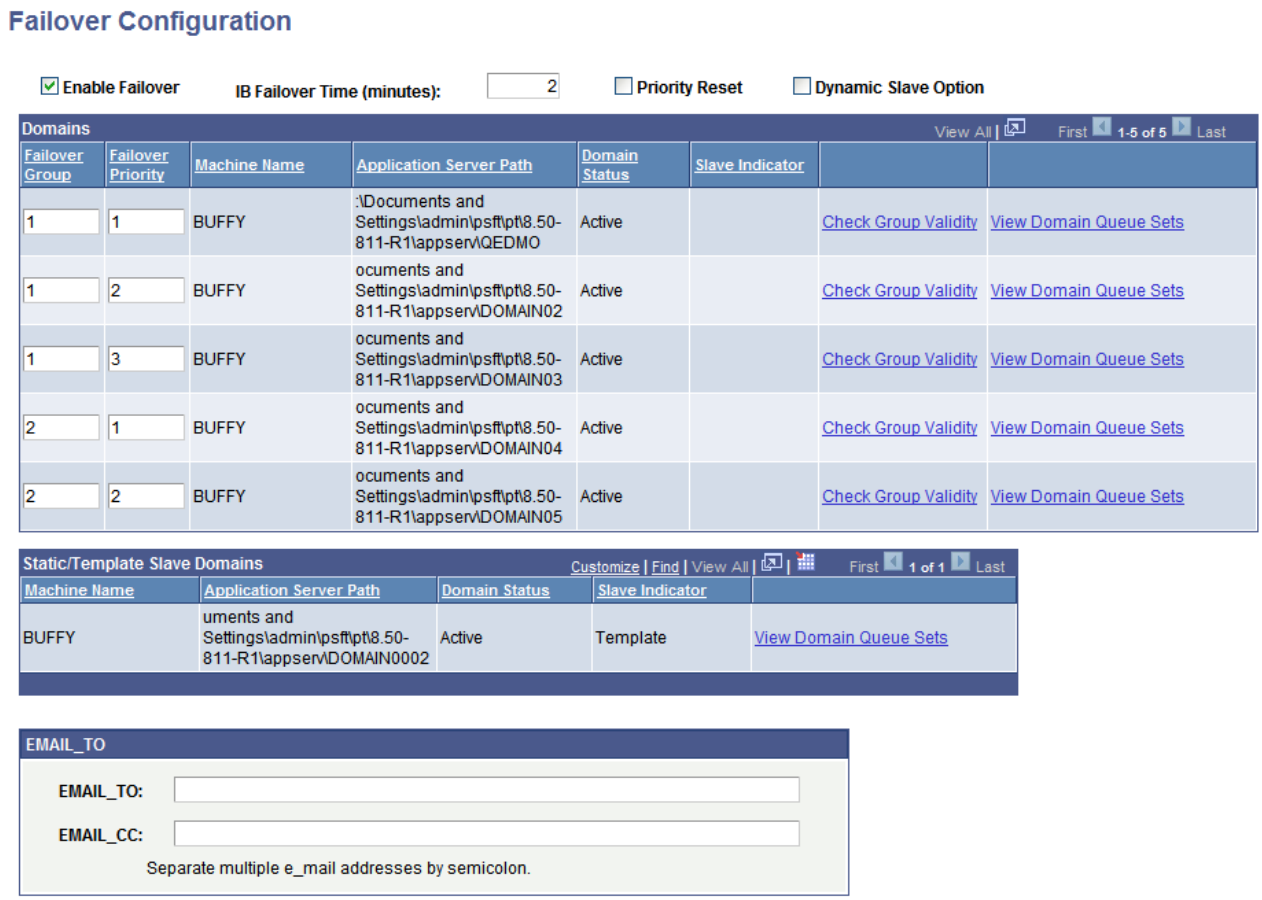
Failover and Node Pause Times

Domain failover is disabled during node/system pause times. Additional information is provided elsewhere PeopleBooks.

See *Enterprise PeopleTools 8.50 PeopleBook: Integration Broker Service Operations Monitor*, "Pausing, Testing, and Pinging Nodes," Understanding Pausing Nodes.

Enabling Failover on Domains

Use the Failover Configuration page (IB_AMM_FAILOVER) to enable failover on domains. The following example shows the page:



Failover configuration page.

To set up domain failover:

1. Access the Failover Configure page (PeopleTools, Integration Broker, Service Operations Monitor, Administration, Domain Status and click the Set Up Failover link).
2. Select the Enable Failover box.
3. In the IB Failover Time (minutes) field, specify the number of minutes that can pass without the domain registering itself before the failover should commence
4. (Optional.) To implement dynamic slaves, select the Dynamic Slave Option.
5. (Optional.) In the Failover Group field, enter a numeric value to specify a group to which a domain belongs.

   A value of *1* indicates that the domain is the first backup domain; a value of *2* indicates that the domain is the second back-up domain if the first backup domain fails; and so on.
6. In the Failover Priority field, enter a numeric value to specify the priority for a back up domain in the failover configuration.

   A value of *1* indicates that the domain is the first backup domain; a value of *2* indicates that the domain is the second back up domain; and so on.

7. (Optional.) In the Email_TO field, specify the email addresses of people to whom the system sends a notification about the domain failure if it occurs.

   Separate multiple email addresses with a semicolon.

8. (Optional.) In the Email_CC field, specify the email addresses of people who receive copies of the domain failure notification.

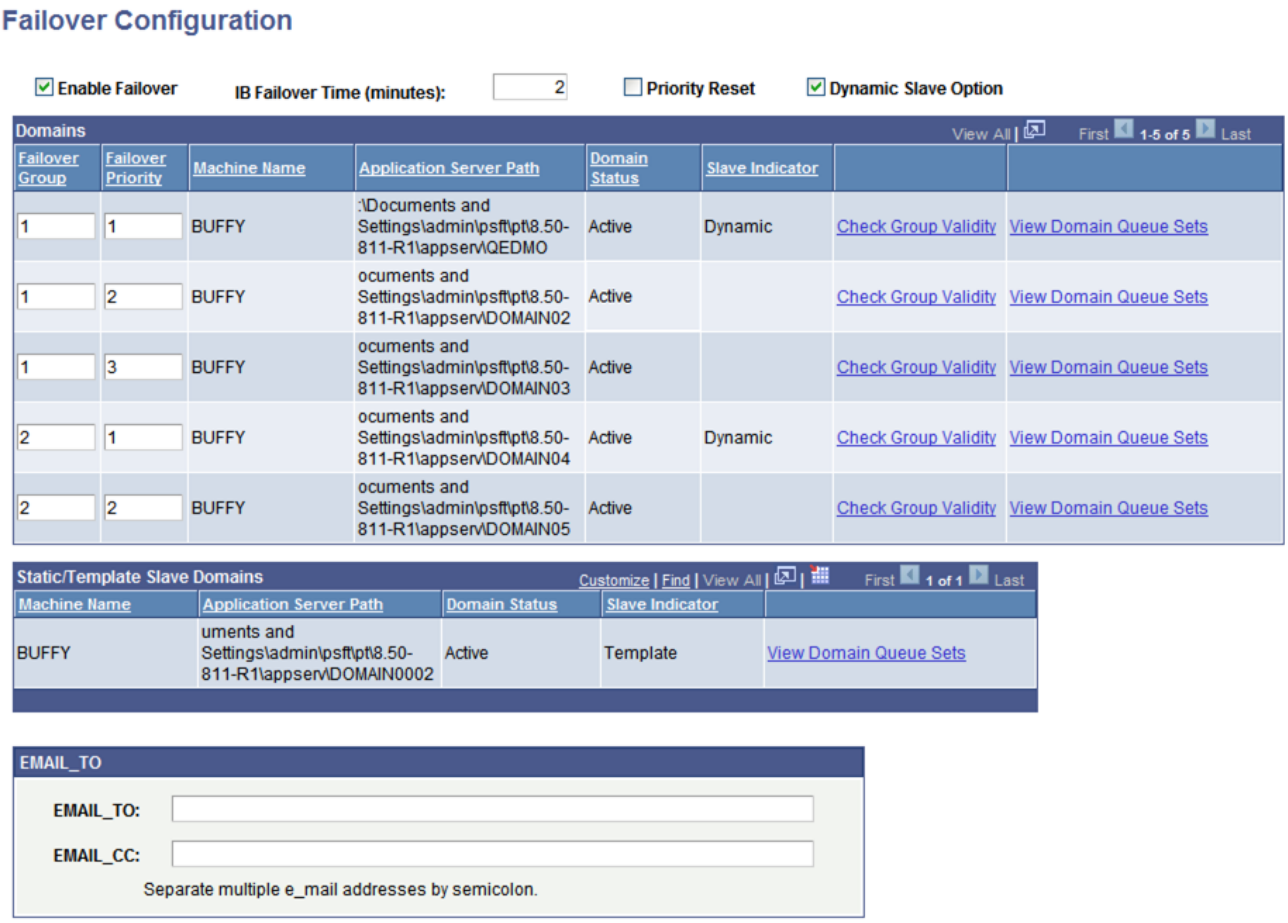
   Separate multiple email addresses with a semicolon.

9. Click the Save button.

Setting Up Dynamic Master-Slave Dispatchers

When dynamic slaves have been set the Slave Indicator column on the Failover Configuration page displays the status *Dynamic* to indicate the domains that are serving as dynamic slaves.

The following example shows the Failover Configuration page with two dynamic slaves in place.



Dynamic slaves configured in the Failover Configuration page

As noted earlier, the domain in a failover group with the highest priority becomes the master and the domain with the second highest priority becomes the slave.

To set up master-slave dispatchers, follow the procedure for setting up failover and verify that you:

- Check the Enable Failover box.
- Check the Dynamic Slave Option box.
- Set up at least one failover group that contains at least two domains.
- Set a failover priority for each domain in the failover group.
- Save your settings.

Checking Queue Validity

Use the Check Group Validity link on the Failover Configuration page to verify that all queues assigned to the pub/sub processes in a failover group are the same.

When you click the link a message box appears that indicates if the group is valid.

Viewing Queues Assigned to Failover Groups

Use the View Group Queues link on the Failover Configuration page to view the queues assigned to each dispatcher in a failover group. Queues must be identical among all groups.

Configuring Integration Gateways for Load Balancing

This section discusses how to configure an integration gateways for load balancing.

Understanding Configuring Integration Gateways for Load Balancing

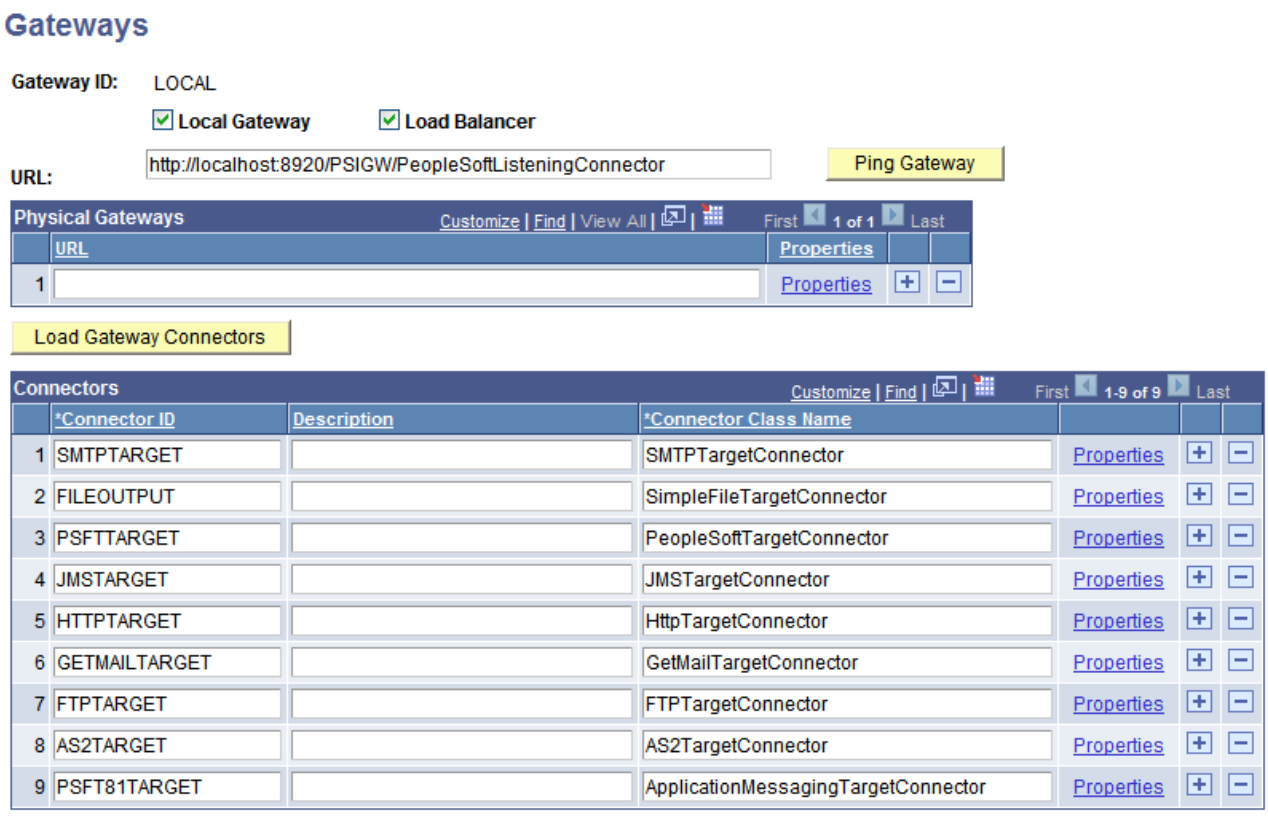
To increase gateway performance you can use load balancing. Load balancing involves the use of a third-party load balancing software product and the installation and configuration of multiple gateways. Then, when messages are sent or published to your messaging system, the load balancing software analyzes the load on installed gateways and determines to which gateway to send the messages to balance the load on all gateways.

For installation and configuration information about your load balancing software, please see the documentation that is included with the product.

Configuring Load Balancing

To configure gateways participating in load balancing, you must specify the URLs of the gateways in use for load balancing on the Gateways page, and then set integration gateway properties for each gateway you specify. Note that you can set different properties for each gateway.

To access the Gateways page, select PeopleTools, Integration Broker, Integration Setup, Gateways. Select the default local gateway.



Gateways page with load balancing enabled

To configure an integration gateway for load balancing:

1. Select the Load Balancer box.
2. In the Physical Gateway section, in the URL field, enter a gateway URL for a gateway that will be used for load balancing.
3. Click the plus (+) button and enter gateway URLs for each additional gateway to be used for load balancing.
4. Click the Save button.
5. For each gateway URL entered, click the Properties link to set integration gateway properties for that gateway.

See Also

Chapter 5, "Managing Integration Gateways," page 29

Implementing Load Balancing on Service Operation Queues

This section discusses implementing load balancing on service operation queues.

Understanding Implementing Load Balancing on Service Operation Queues

PeopleSoft provides the ability to load balance queue processing on active pub/sub systems using the Load Balance Interval parameter in PSAdmin.

Load Balancing on Pub/Sub Systems

PeopleSoft provides the ability to load balance queue processing on active pub/sub systems using the Load Balance Interval parameter in PSAdmin.

Note. The Load Balance Interval parameter is the same parameter used to resubmit failed transactions.

For example, without the Load Balance Interval parameter set, if there are three queues that have service operations to process, only one queue gets processed at a time. Moreover, as TPA calls continue to be generated, the dispatcher looks in the same queue to process service operations, resulting in that single queue performing most of the processing. This scenario happens most frequently when publishing in batch mode. As long as there are service operations in that one queue, the system does not process any service operations in any other queue.

However, when you set the Load Balance Interval parameter and the value you set is exceeded, the system dispatches all queues. This means that other queues that can be processed will be processed, at least partially, for the interval time designated.

Multi-Queue Processing

Multi-queue processing enables you to assign multiple queues for unordered service operation transaction processing.

When you configure multi-queue processing, a service operation transaction are processed one at a time on a rotating basis among all queues defined for multi-queue processing. You can also use this feature for

See Also

Chapter 4, "Administering Messaging Servers for Asynchronous Messaging," Specifying Dispatcher Parameters, page 23

Implementing Load Balancing on Pub/Sub Systems

This section discusses load balancing on pub/sub systems.

Understanding Load Balancing on Pub/Sub Systems

You can implement load balancing on pub/sub systems by setting a value, in minutes, that determines the time interval between dispatcher processing of requests.

When implementing, processing consists of the system attempting to perform the equivalent of an on idle ping on all down nodes (without the ping interval delay) for the default publication contract dispatcher (_dflt). Moreover, for all dispatchers not configured as slaves when this interval is reached, the equivalent of on idle processing will be performed. This means that if other queues can be processed they will be for the current dispatch cycle.

Setting the Load Balance Interval Parameter

The Load Balance Interval parameter is located in the Settings for Pub/Sub Servers section of PSAdmin. By default this feature is disabled and has a default setting of *0.* The value you set is the number of minutes between load balance processing.

See Also

Enterprise PeopleTools 8.50 PeopleBook: System and Server Administration, "Using the PSADMIN Utility"

Resubmitting Failed Transactions

The PSADMIN parameter Load Balance Interval enables you to resubmit failed transactions for processing.

This functionality allow transactions that failed due to a connection problem to be retried periodically. The benefit of this is to unblock a queue and have it be able to process in a more load balancing way.

Note. The Load Balance Interval parameter is the same parameter used to implement load balancing on service operation queues.

The Load Balance Interval parameter is located in the Settings for Pub/Sub Servers section of PSAdmin. The value you set for the Load Balance Interval parameter determines the time interval (in minutes) between load balance processing when the dispatcher is processing requests.

When this parameter is enabled, processing consists of attempting to perform the equivalent using the Scan Interval parameter, without the delay. Moreover, when this load balance interval is reached the equivalent of the scan interval processing is performed on all default dispatchers, allowing other queues to process transactions.

When true scan interval processing is performed the load balance interval time is reset.

Note. Only the default publication contract dispatcher (_dflt) is used to ping these nodes. When the load balance interval is exceeded the default publication contract dispatcher performs the scan interval processing of pinging the nodes. If the actual scan interval processing is run before the load balancing interval is exceeded, then the system resets the load balance time value.

See Also

Chapter 4, "Administering Messaging Servers for Asynchronous Messaging," Specifying Dispatcher Parameters, page 23

Chapter 12, "Tuning Messaging System Performance," Setting the Load Balance Interval Parameter, page 264

Using the Bulk Load Handler for Large Message Subscriptions

PeopleSoft Integration Broker provides a bulk load handler type that serves as a bulk loader to insert data. This handler is available when working with asynchronous one-way service operation types.

See *Enterprise PeopleTools 8.50 PeopleBook: PeopleSoft Integration Broker*, "Managing Service Operation Handlers," Implementing Handlers Using Bulk Load Processing.

Managing Pub/Sub Process Handler Performance

This section discusses how to:

- Enable the serial recycling of pub/sub process handlers.
- Recycle pub/sub process handlers based on process memory growth.

Enabling Serial Recycling of Pub/Sub Process Handlers

When serial recycle for pub/sub process handlers is enabled, the system recycles process handlers (within a group) on a serial basis–one after another–to allow processing to continue uninterrupted.

If serial recycling is disabled, all pub/sub process handlers recycle at once, which can cause throughput to come to a standstill.

By default the serial recycling of pub/sub process handlers is enabled.

Serial recycling uses the following parameters that you set in the psappsrv.cfg file in the Settings for PUB/SUB Servers section:

Serial Recycle To enable serial recycling enter *Y*. To disable serial recycling enter *N*.

The default value is *Y*.

Serial Recycle Time Specifies the maximum interval in seconds at which the system recycles a process.

The minimum valid recycle time is 60 seconds

The default value is *60* seconds.

To enable serial recycling, uncomment the parameters in the psappsrv.cfg file and set the appropriate values. After you have made your changes, save the file and reboot the application server.

To disable serial recycling, comment out the parameters, save the file and reboot the application server.

Recycling Pub/Sub Process Handlers Based on Process Memory Growth

PeopleSoft Integration Broker enables you to recycle pub/sub process handlers based on memory growth in cache.

You use the Percentage of Memory Growth parameter in the psappsrv.cfg file to specify that the system recycle pub/sub process handlers when memory has grown by a value you specify. The system checks to see if the percentage criterion is met after processing a specified number of requests.

By default the Percentage of Memory Growth parameter is disabled.

This feature uses the following parameters that you set in the psappsrv.cfg file in the Settings for PUB/SUB Servers section:

Percentage of Memory Growth Specifies the percentage growth of memory in cache at which the system recycles pub/sub process handlers.

The default value is *20* percent.

Interval Parameter Determines the number of requests for the system to process before checking the percent memory growth in cache.

The default value is *100.*

To enable process handler recycling based on memory growth, uncomment the parameters in the psappsrv.cfg file and set the appropriate values. After you have made your changes, save the file and reboot the application server.

To disable process handler recycling based on memory growth, comment out the parameters, save the file and reboot the application server.

Appendix A

Using the Delivered Listening Connectors and Target Connectors

This appendix provides examples for using the listening and target connectors delivered with PeopleSoft Integration Broker, and discusses how to:

- Set up metadata for the examples.
- Use the PeopleSoft connectors.
- Use the HTTP connectors.
- Use the PeopleSoft 8.1 connectors.
- Use the JMS connectors.
- Use the AS2 connectors.
- Use the simple file target connector.
- Use the FTP target connector.
- Use the SMTP target connector.

Understanding Using This Appendix

This appendix presents examples of how to use the connectors delivered with PeopleSoft Integration Broker.

The intention of the examples provided in this appendix is to provide a starting point for exploring how the connectors work. The examples are designed to be simple and require the minimum set up and configuration necessary to invoke them.

If you try these examples and choose to cut the code samples provided in this document and paste them into PeopleSoft Application Designer, the PeopleSoft Pure Internet Architecture, or text or XML editors, verify that single or double quotation marks are pasted into these mediums as straight quotes. Slanted or curly quotes will cause the code samples to fail.

Prerequisites

To use this appendix, you should have basic experience in using PeopleSoft Integration Broker. There is little background information presented in this appendix and many of the basic steps involved in creating integrations are presented in general terms (for example, "create a new Service/Service Operation.") Please refer to the appropriate chapters in this PeopleBook for information on how to complete basic tasks.

Setting Up Metadata

This section discusses how to set up metadata for the examples presented in this appendix and discusses how to:

- Create queues, request messages, response messages, services, service operations and pages.
- Create nodes and routing definitions.
- Create a test record and page.
- Set up integration gateway logging.

Understanding Setting Up Metadata

Before you use the examples in this appendix you must set up metadata as described in this section.

Note. The examples presented in this appendix demonstrate the use of one type of connector at a time. The examples share the same basic definitions for the service operation, request message, response message, routings, and the test page. As a result, you should attempt to run only one example at a time, since the underlying metadata and objects are shared.

The exact requirements for setting up the listening and target connectors do differ somewhat, but since the differences are fairly minor the steps are combined in this section.

Prerequisites

Before you begin the set up data for the examples configure and start the integration gateway.

Creating Services, Service Operations, Queues, and Messages

This section describes creating services, service operations, queues, and request and response messages for use in running the connector examples presented in this appendix.

Unless otherwise noted, use the appropriate PeopleSoft Pure Internet Architecture pages to complete these tasks.

To create services, service operations, queues, and messages:

1. Create a new Service

   Name the service EXAMPLE_SERVICE.

2. Create new synchronous Service operation.

    a. Add a service operation of type synchronous to the EXAMPLE_SERVICE service and name it *EXAMPLE_SERVICE_OPR.*

    b. Complete the field definitions for service operation as follows:

Field	***Value***
Operation Description	Test service operation
Request Message.Version	*EXAMPLE_REQUEST_MSG.VERSION_1*
Response Message Name.Version	*EXAMPLE_RESPONSE_MSG.VERSION_1*

    c. Configure the Service Operation Security for this service operation.

3. Create a new asynchronous Service Operation

    a. Add a service operation of type Asynchronous — one way to the EXAMPLE_SERVICE and name it *EXAMPLE_SERVICE_ASYNC_OPR*

    b. Complete the field definitions for the service operation as follows:

Field	***Value***
Operation Description	Test service operation
Request Message.Version	*EXAMPLE_REQUEST_MSG.VERSION_1*
Queue Name	EXAMPLE_QUEUE

    c. Configure the Service Operation Security for this service operation.

4. Create a new queue.

    a. Name the queue *EXAMPLE_QUEUE*

    b. Verify that the Queue Status is set to Run.

    c. Use the Integration Broker Service Operations Monitor Administration to verify that the *EXAMPLE_QUEUE* is running.

5. Create a new request message.

    Create a Nonrowset-based message with message name as *EXAMPLE_REQUEST_MSG* and message version as *VERSION_1*.

6. Create a new response message.

    Create a Nonrowset-based message with message name as *EXAMPLE_RESPONSE_MSG* and message version as *VERSION_1*.

Creating the Test Record and Page.

This section discusses how to use PeopleSoft Application Designer to:

- Create a test record.
- Create a test page.

Creating the Test Record

You must create a work record that will be used on the Test Page.

Create a new record:

1. Insert the character field *TEST* into the record.
2. Select Derived/Work as the Record Type.
3. Save the record as *EXAMPLE_WORKREC.*

Creating the Test Page

You must create a test page. This page will be used in some of the target connector examples.

Create a new page with a single push button on it:

1. Create the page.
2. Add a push button with the following properties:

Property	***Value***
Destination	PeopleCode Command
RecordName	*EXAMPLE_WORKREC*
Field Name	*TEST*

3. Re-size the button and label it *Test target connector.*
4. Save the page as *EXAMPLE_PAGE.*
5. Add the page to a component. This may be an existing component or a new one. Ensure that the security settings for the component allow the new page to be accessed.

Creating Nodes and Routing Definitions

Use the PeopleSoft Pure Internet Architecture to complete the following tasks.

Creating Source Nodes and Inbound Routing for service operations

You must create a node that will be the source of all requests to the listening connectors.

To create a source node and a inbound routing:

1. Add a new node called *SOURCENODE.* Enter in appropriate values for the description and the default user ID. Verify that the Active Node check box has been selected. Save this node.
2. Add a new inbound routing to the *EXAMPLE_SERVICE_OPR* service operation and name it EXAMPLE_SERVICE_IN_RTN.
    a. Set the Sender Node field value to SOURCENODE and the Receiver Node field value to the local node's value.
    b. Check the *Active* check-box for routing .
    c. Set the Logging Details field value to Header and Detail .
    d. Save the routing.

Adding Target Nodes and outbound routing

You must create a target node and an outbound routing for all outgoing requests for the target connectors.

To add a target node and an outbound routing:

1. Add a new node called *TARGETNODE.* Enter in the appropriate values for the description and default user ID. Verify that the Active Node check box has been selected. Save this node.
2. Add a new outbound routing to the *EXAMPLE_SERVICE_OPR* service operation and name it EXAMPLE_SERVICE_OUT_RTN.
    a. Set the Sender Node field value to the local node's value and the Receiver Node field value to TARGETNODE.
    b. Verify that the Status is set to Active.
    c. Verify that Logging Details field value is set to Header and Detail.
    d. Save the routing.
3. Add a new outbound routing to the service operation *EXAMPLE_SERVICE_OPR_ASYNC* and name it EXAMPLE_SERVICE_ASYNC_RTN.
    a. Set the Sender Node field value to the local node's value and the Receiver Node field value to TARGETNODE.
    b. Verify that the Status is set to Active.
    c. Verify that Logging Details field value is set to Header and Detail.
    d. Save the routing.

Setting Up Integration Gateway Logging

The integration gateway has message and error logging capabilities. If problems arise while trying the examples, these logs can be invaluable in determining where problems are occurring.

See *Enterprise PeopleTools 8.50 PeopleBook: PeopleSoft Integration Broker*, "Managing Error Handling, Logging, Tracing, and Debugging," Managing Integration Gateway Message and Error Logging.

Example 1: Using the PeopleSoft Connectors

This section discusses using the PeopleSoft listening and PeopleSoft target connectors.

Understanding the PeopleSoft Connector Examples

The example provided for using the PeopleSoft target connector demonstrates using the connector to invoke a synchronous service operation between two PeopleSoft nodes.

The example provided for using the PeopleSoft listening connector demonstrates using Send Master to invoke a service operation into the local system for processing.

Prerequisites

To use the PeopleSoft target connector example you must have a second PeopleSoft 8.50 system. You must have the application server, the PeopleSoft Pure Internet Architecture and the Integration Gateway configured and running.

Note. In this section, the current PeopleSoft system is referred to as the *originating* system, and the second PeopleSoft system is called the *destination* system.

Using the PeopleSoft Target Connector

This section provides an example of using the PeopleSoft target connector and describes how to:

- Set up data on the originating system.
- Set up data on the destination system.
- Test the PeopleSoft target connector.

Setting Up Data on the Originating System

To set up data on the originating system:

1. In PeopleSoft Application Designer, open the EXAMPLE_WORKREC record. Add the following PeopleCode to the FieldChange event for the TEST field:

```
&msg = CreateMessage(Operation.EXAMPLE_SERVICE_OPR);

&xmldata = "<?xml version='1.0'?><ConnectorTest/>";

/* create an XmlDoc */
&xmlDoc = CreateXmlDoc(&xmldata);
&rootNode = &xmlDoc.documentelement;
&descNode = &rootNode.addelement("PSFTtest");
&descNode.nodevalue = "This is a test message.";

/* put the XML in the message */
&msg.setxmldoc(&xmlDoc);

/* send the request */
&response = %IntBroker.SyncRequest(&msg);

/* and echo it back to the user */
&xmlDoc = &response.getxmldoc();
MessageBox(0, "", 0, 0, &xmlDoc.genxmlstring());
```

2. In the PeopleSoft Pure Internet Architecture, open the node definition for TARGETNODE. Set the ConnectorID to PSFTTARGET.

3. In the Integration Properties for the gateway, add a new entry for TARGETNODE along with the appropriate values.

```
ig.isc.TARGETNODE.serverURL=//<machinename>:<port>
ig.isc.TARGETNODE.userid=<userid>
ig.isc.TARGETNODE.password=<password>
ig.isc.TARGETNODE.toolsRel=<toolsRelease>
```

Setting Up Data on the Destination System

To set up data on the destination system:

1. Follow the steps outlined in the section "Setting Up Metadata" to add the following to the destination system:

   a. the EXAMPLE_QUEUE queue

   b. the EXAMPLE_REQUEST_MSG message

   c. the EXAMPLE_RESPONSE_MSG message

   d. the EXAMPLE_SERVICE service

   e. the EXAMPLE_SERVICE_OPR synchronous service operation

2. Add a node entry for the originating system. Ensure that the Single Signon security is configured so that the destination system accepts authentication tokens from the originating system.

3. Add a new inbound synchronous routing between the originating system and the destination for the EXAMPLE_SERVICE_OPR service operation.

4. In PIA, for service operation *EXAMPLE_SERVICE_OPR* add a handler of type OnRequest with implementation type App Class. Create a handler application class based on the IRequestHandler interface, and for the method OnRequest add following PeopleCode

```
Local XmlDoc &xmldoc;
Local File &theFile;
Local XmlNode &rootNode, &descNode;
Local Message &response;
Local string &xmldata;

/* get the body of the incoming message */
&xmldoc = &_MSG.GetXmlDoc();

/* and write it out to a file */
&theFile = GetFile("ARequest.txt", "W");
&theFile.WriteString(&xmldoc.GenXmlString());
&theFile.Close();

/* create the response message */
&response = CreateMessage(Operation.EXAMPLE_SERVICE_OPR, %IntBroker_Response);

/* create the body for the response message */
&xmldata = "<?xml version='1.0'?><ConnectorTest/>";
&xmldoc = CreateXmlDoc(&xmldata);
&rootNode = &xmldoc.DocumentElement;
&descNode = &rootNode.AddElement("ResponseMessage");
&descNode.NodeValue = "This was generated in the OnRequest event.";

/* add the body to the message */
&response.SetXmlDoc(&xmldoc);

/* and return the response message */

Return &response;
```

Testing the PeopleSoft Target Connector

To test the PeopleSoft target connector:

1. In the PeopleSoft Pure Internet Architecture, open the EXAMPLE_PAGE page and click the Test button. The response message will be displayed in a message box.

2. On the destination system, open Service Operation Monitor to view the details of the received message. Open the text file created by the OnRequest PeopleCode to view the details of service operation request received.

Using the PeopleSoft Listening Connector

This section provides an example for testing the PeopleSoft listening connector.

Testing the PeopleSoft Listening Connector

To test the PeopleSoft listening connector:

1. In PIA , open the EXAMPLE_SERVICE_OPR service operation and add a Handler of type OnRequest with implementation type App class. The OnRequest method of App class should have following PeopleCode .

```
Local XmlDoc &xmldoc;
Local File &theFile;
Local XmlNode &rootNode, &descNode;
Local Message &response;
Local string &xmldata;

/* get the body of the incoming message */
&xmldoc = &_MSG.GetXmlDoc();

/* and write it out to a file */
&theFile = GetFile("HttpRequest.txt", "W");
&theFile.WriteString(&xmldoc.GenXmlString());
&theFile.Close();

/* create the response message */
&response = CreateMessage(Operation.EXAMPLE_SERVICE_OPR, %IntBroker_Response);

/* create the body for the response message */
&xmldata = "<?xml version='1.0'?><ConnectorTest/>";
&xmldoc = CreateXmlDoc(&xmldata);
&rootNode = &xmldoc.DocumentElement;
&descNode = &rootNode.AddElement("ResponseMessage");
&descNode.NodeValue = "This was generated in the OnRequest event.";

/* add the body to the message */
&response.SetXmlDoc(&xmldoc);

/* and return the response message */

Return &response;
```

2. Start Send Master and create an 8.48 Integration Broker (MIME) project.
3. In the URL field enter the address of the PeopleSoft listening connector:

   ```
   http://your_server_name/PSIGW/PeopleSoftListeningConnector
   ```

   Replacing <your_server_name> with the details of the server where the gateway is running. For example:

   ```
   http://machine1234/PSIGW/PeopleSoftListeningConnector
   ```

4. In the Requesting Node field, enter *SOURCENODE.*
5. In the Ext. Operation name field, enter *EXAMPLE_SERVICE_OPR.v1*.
6. From the Operation type list, select Sync.
7. Click the Input File tab and enter the following XML:

   ```
   <?xml version="1.0"?><Test>Data</Test>
   ```

8. Click the Post button.

   The response from the server displays in the Output Information section. Note that this is a MIME response; look near the end to find the response XML generated by the OnRequest PeopleCode. Open the text file created by the OnRequest method of application class to view the body of the request message.

Example 2: Using the HTTP Connectors

This section discusses how to:

- Use the HTTP listening connector.
- Use the HTTP target connector.

Prerequisites

When using the examples for using the HTTP target connector, an HTTP server is needed to receive the HTTP request and to return a response. If using the SOAP example, the HTTP server must be able to process SOAP messages.

Using the HTTP Listening Connector

This section provides examples of how to set credentials for HTTP requests coming into the integration gateway, and discusses how to:

- Set credentials in message bodies.
- Set credentials in HTTP headers.
- Set credentials in query strings.
- Set credentials in SOAP-specific HTTP headers.

Setting Up for Using the HTTP Listening Connector Examples

In PIA, for service operation *EXAMPLE_SERVICE_OPR* add a handler of type OnRequest with implementation type application Class. Create a handler application class based on the IRequestHandler interface, and for the method OnRequest add following PeopleCode


```
Local XmlDoc &xmldoc;
Local File &theFile;
Local XmlNode &rootNode, &descNode;
Local Message &response;
Local string &xmldata;

/* get the body of the incoming message */
&xmldoc = &_MSG.GetXmlDoc();

/* and write it out to a file */
&theFile = GetFile("HttpRequest.txt", "W");
&theFile.WriteString(&xmldoc.GenXmlString());
&theFile.Close();

/* create the response message */
&response = CreateMessage(Operation.EXAMPLE_SERVICE_OPR,
  %IntBroker_Response);

/* create the body for the response message */
&xmldata = "<?xml version='1.0'?><ConnectorTest/>";
&xmldoc = CreateXmlDoc(&xmldata);
&rootNode = &xmldoc.DocumentElement;
&descNode = &rootNode.AddElement("ResponseMessage");
&descNode.NodeValue = "This was generated in the OnRequest event.";

/* add the body to the message */
&response.SetXmlDoc(&xmldoc);

/* and return the response message */
Return &response;
```

Setting Credentials in the Message Body

To set HTTP request credentials in the message body:

1. Start Send Master, and create a new Input File project.
2. In the URL field enter:

   ```
   http://<your_server_name>/PSIGW/HttpListeningConnector
   ```

   Replace <your_server_name> with the details of the server where the integration gateway is running. For example:

   ```
   http://machine1234/PSIGW/HttpListeningConnector
   ```

3. In the Input section, paste the following XML. Notice that the service operation name and requesting node are present in the XML

```
<?xml version="1.0"?>
<IBRequest>
  <ExternalOperationName>EXAMPLE_SERVICE_OPR.v1</ExternalOperation
   Name>
  <From>
    <RequestingNode>SOURCENODE</RequestingNode>
 </From>
 <ContentSections>
   <ContentSection>
     <Data>
       <![CDATA[<?xml version="1.0"?><ConnectorTest>
       Testing the HTTPListeningConnector. Message body.
       </ConnectorTest>]]>
     </Data>
   </ContentSection>
 </ContentSections>
</IBRequest>
```

4. Click the Post button to invoke service operation on the integration gateway.

5. Check the Output section for the response. Compare the response with the XML created in the handler application class. Also check the HttpRequest.txt file created by the OnRequest PeopleCode to see the body of the request message received by the application server.

Setting Credentials in HTTP Headers

To set HTTP request credentials in the HTTP header:

1. Start Send Master, and create a new Input File project.

2. In the URL field enter:

   ```
   http://<your_server_name>/PSIGW/HttpListeningConnector
   ```

   Replace <your_server_name> with the details of the server where the integration gateway is running. For example:

   ```
   http://machine1234/PSIGW/HttpListeningConnector
   ```

3. In the Headers field enter the following:

   ```
   OperationName:EXAMPLE_SERVICE_OPR.v1
   From:SOURCENODE
   ```

4. In the Input section, paste the following:

   ```
   <?xml version="1.0"?>
   <ConnectorTest>
   Testing the HTTPListeningConnector. HTTP Header.
   </ConnectorTest>
   ```

5. Click the Post button to sent the message to the integration gateway.

6. Check the Output section for the response. Compare the response with the XML created in the handler application class. Also check the HttpRequest.txt file created by the OnRequest PeopleCode to see the body of the request message received by the application server.

Setting Credentials in Query Strings

To set HTTP request credentials in a query string:

1. Start Send Master, and create a new Input File project.
2. In the URL field enter:

   ```
   http://your_server_name/PSIGW/HttpListeningConnector?&Operation=
   EXAMPLE_SERVICE_OPR.v1&From=SOURCENODE
   ```

   Replace <your_server_name> with the details of the server where the integration gateway is running. For example:

   ```
   http://machine1234/PSIGW/HttpListeningConnector?&Operation=
   EXAMPLE_SERVICE_OPR.VERSION_1&From=SOURCENODE
   ```

3. In the Input section, paste the following:

   ```
   <?xml version="1.0"?>
   <ConnectorTest>
   Testing the HTTPListeningConnector. Query String.
   </ConnectorTest>
   ```

4. Click the Post button to invoke service operation on the integration gateway.
5. Check the Output section for the response. Compare the response with the XML created in the handler application class. Also check the HttpRequest.txt file created by the OnRequest PeopleCode to see the body of the request message received by the application server.

Setting Credentials in SOAP-Specific HTTP Headers

To set HTTP request credentials in a SOAP-specific HTTP header:

1. Start Send Master, and create a new Input File project.
2. In the URL field enter:

   ```
   http://your_server_name/PSIGW/HttpListeningConnector
   ```

   Replacing <your_server_name> with the details of the server where the gateway is running. For example:

   ```
   http://machine1234/PSIGW/HttpListeningConnector
   ```

3. In the Header field, add the following:

   ```
   SOAPAction: http://peoplesoft.com/EXAMPLE_SERVICE_OPR.v1/SOURCENODE//
   ```

4. In the Input section, paste the following:

```
<?xml version="1.0"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENC="http://schemas.xmlsoap.encoding/
 "xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Body>
      <ConnectorTest>
         <Text>
           Testing the HTTPListeningConnector. SOAP Message.
         </Text>
      </ConnectorTest>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

5. Click the Post button to invoke service operation on the integration gateway.

6. Check the Output section for the response. Compare the response with the XML created in the handler application class; that XML will be returned wrapped in a SOAP envelope. Also check the HttpRequest.txt file created by the OnRequest PeopleCode to see the body of the request message received by the application server.

Using the HTTP Target Connector

This section provides examples of using the HTTP target connector and discusses how use the connector to:

- Send standard HTTP requests.
- Send SOAP messages in HTTP requests.

Sending Standard HTTP Requests

To send a standard HTTP request:

1. In PeopleSoft Application Designer, open the EXAMPLE_WORKREC record and add the following PeopleCode to the FieldChange event for the TEST field.

```
&msg = CreateMessage(Operation.EXAMPLE_SERVICE_OPR);

&xmldata = "<?xml version='1.0'?><ConnectorTest/>";

/* create an XmlDoc */
&xmlDoc = CreateXmlDoc(&xmldata);
&rootNode = &xmlDoc.documentelement;
&descNode = &rootNode.addelement("HTTPtest");
&descNode.nodevalue = "This will be sent to an HTTP server.";

/* put the XML in the message */
&msg.setxmldoc(&xmlDoc);

/* send the request */
&response = %IntBroker.SyncRequest(&msg);

/* and echo it back to the user */
&xmlDoc = &response.getxmldoc();
MessageBox(0, "", 0, 0, &xmlDoc.genxmlstring());
```

   Note that this code assumes that the response from the server is properly formatted XML.

2. In the PeopleSoft Pure Internet Architecture, open the node definition for TARGETNODE. Set the Connector ID to *HTTPTARGET*. Set the URL property value to the address of the HTTP server that will process the request.

3. Open the EXAMPLE_PAGE page, and click on the Test button. The HTTP response will be displayed in the resulting message box.

Sending SOAP Messages in HTTP Requests

To send a SOAP message in an HTTP request:

1. In PeopleSoft Application Designer, open the EXAMPLE_WORKREC record and add the following PeopleCode to the FieldChange event for the TEST field.

```
&msg = CreateMessage(Operation.EXAMPLE_SERVICE_OPR);

/* create a SOAP document */
&soapReq = CreateSOAPDoc();

&soapReq.AddMethod("TestNode", 1);
&soapReq.AddParm("Text", "This is a SOAP request.");

/* put the XML in the message */
&msg.setxmldoc(&soapReq.xmlDoc);

/* send the request */
&response = %IntBroker.SyncRequest(&msg);

/* and echo it back to the user */
&xmlDoc = &response.getxmldoc();
MessageBox(0, "", 0, 0, &xmlDoc.genxmlstring());
```

2. In the PeopleSoft Pure Internet Architecture, open the node definition for TARGETNODE.

   a. On the Node Definitions-Connectors tab, set the Connector ID to *HTTPTARGET*.

   b. Set the URL property value to the address of the HTTP server that will process the request.

3. Open the EXAMPLE_PAGE page, and click on the Test button. The HTTP response will be displayed in the resulting message box.

Example 3: Using the PeopleSoft 8.1 Connectors

The examples provided in this section demonstrate sending a rowset-based asynchronous message between a PeopleSoft 8.50 node and a PeopleSoft 8.1 node.

Understanding the PeopleSoft 8.1 Connectors Examples

When sending a message from a PeopleSoft 8.50 system to a PeopleSoft 8.1 system, you will use the PeopleSoft 8.1 target connector. You will also use PeopleCode, as well as the example page and work record that you created using the information in the setup section at the beginning of this appendix.

When sending a message from a PeopleSoft 8.1 system to a PeopleSoft 8.4 system, you will use the PeopleSoft 8.1 listening connector. You will also use the test message functionality in PeopleSoft Application Designer.

Setting Up Data for the PeopleSoft 8.1 Connectors Examples

This section describes setting up data for using the PeopleSoft 8.1 connector examples.

Setting Up Data on the PeopleSoft 8.50 System

To set up data on the PeopleSoft 8.50 system:

1. In PeopleSoft Application Designer, create a new field called *EXAMPLE_CHAR*. This should be a mixed-case character field of size *20*.
2. Create a new record.
   a. Name the record *EXAMPLE_REC*.
   b. Add the *EXAMPLE_CHAR* field to this record, set it as the key, and save the definition.
   c. Build the physical table for this record.
3. In the PeopleSoft Pure Internet Architecture, create a new message called EXAMPLE_PSFT_MSG with the version set to VERSION_1.
   a. Select the message type to be Rowset— Based.
   b. Add the *EXAMPLE_REC* record as the root record of this message.
4. Add a new node, using the node name of the PeopleSoft 8.1 system. Verify that the Active Node box is checked, and save the record.
5. Open the EXAMPLE_PAGE page and add an EditBox to the page, setting the following properties:

Property	***Value***
Record name	*EXAMPLE_REC*
Field name	*EXAMPLE_CHAR*

6. Create a new service called PSFT81_SERVICE.
7. Create a new service operation
   a. Add a service operation of type asynchronous-one way to the PSFT81_SERVICE and name it PSFT81_SERVICE_OPR.
   b. Add EXAMPLE_PSFT_MSG as the message.
   c. Add EXAMPLE_QUEUE as the queue.
   d. Configure the Service Operation Security for this service operation.

8. Add an inbound routing for the PSFT81_SERVICE_OPR service operation with the sourcenode being the 8.1 system and the destination being the 8.50 system.

9. Add an outbound routing for the PSFT81_SERVICE_OPR service operation with the sourcenode being the 8.50 system and the destination being 8.1 system.

10. Open the *EXAMPLE_WORKREC* record. Add the following PeopleCode to the FieldChange event for the TEST field:

```
&message = CreateMessage(Operation.PSFT81_SERVICE_OPR);

/* get the buffer data */
&rowset = GetLevel0();

/* copy buffer data to the message */
&message.CopyRowset(&rowset);

/* send the message */
&message.Publish();
```

11. Go to the connector information for the new node. Set the Connector ID to *PSFT81TARGET*. Set the URL property to the address of the gateway servlet on the PeopleSoft 8.1 system. For example:

```
http://<theServerNameAndPort>/servlets/gateway
```

Setting Up Data on the PeopleSoft 8.1 System

To set up data on the PeopleSoft 8.1 system:

1. In PeopleSoft Application Designer, create a new field called *EXAMPLE_CHAR*. This should be a mixed-case character field of size *20*.

2. Create a new record.

   a. Name the record *EXAMPLE_REC*.

   b. Add the *EXAMPLE_CHAR* field to this record, set it as the key, and save the definition.

   c. Build the physical table for this record.

3. Create a new message channel called *EXAMPLE_CHANNEL*. On the properties dialog box, set the Status to Run. Configure the security for the message monitor so that the channel can be displayed.

4. Create a new message.

   a. Open the properties and select the Active box for the Status.

   b. Set the Message Channel to EXAMPLE_CHANNEL.

   c. Add the *EXAMPLE_REC* record to *VERSION_1* of this message.

   d. Save the message as *EXAMPLE_PSFT_MSG*.

5. Add the subscription *ExampleSubscription* to the *EXAMPLE_PSFT_MSG*. Use the following PeopleCode in the subscription body:

```
/* get the incoming message */
&msg = GetMessage();
&msgXML = &msg.GenXMLString();

/* and write it to a file */
&file = GetFile("PSFT81msg.txt", "w");
&file.writeString(&msgXML);
&file.close();
```

6. Create a new message node, using the name of the PeopleSoft 8.50 node. Add a Location to this node with the following format:

   ```
   http://<serverName:port>/PSIGW/PS81ListeningConnector
   ```

   The serverName and port you specify must correspond to the integration gateway address of the PeopleSoft 8.50 system.

7. Open the *EXAMPLE_CHANNEL*. Add a new routing rule to the channel, where the direction is Both and the message node name is that of the PeopleSoft 8.50 node.

8. In the PeopleSoft Pure Internet Architecture, invoke the Gateway Administration servlet and add the PeopleSoft 8.1 node to the PeopleSoft handler.

9. Open the Message Monitor and verify that the *EXAMPLE_CHANNEL* is running.

Using the PeopleSoft 8.1 Target Connector

In the example presented in this section, you will use the PeopleSoft 8.1 target connector to send a message from a PeopleSoft 8.50 system to a PeopleSoft 8.1 system.

To send a message from a PeopleSoft 8.50 system to a PeopleSoft 8.1 system:

1. In the PeopleSoft Pure Internet Architecture on the PeopleSoft 8.50 system, open the EXAMPLE_PAGE. Enter text into the edit box, and press the Test button. Wait for a minute or two to allow the systems to process the message.

2. On the PeopleSoft 8.50 system, open Integration Broker Monitor to view the details of the outbound message.

3. On the PeopleSoft 8.1 system, open up the Message Monitor to view the details of the received message. Open the PSFT81msg.txt file created by the subscription PeopleCode to see the body of the message.

Using the PeopleSoft 8.1 Listening Connector

In the example presented in this section, you will use the PeopleSoft 8.1 listening connector to send a message from a PeopleSoft 8.1 system to a PeopleSoft 8.50 system.

To send a message from a PeopleSoft 8.1 system to a PeopleSoft 8.50 system:

1. On the PeopleSoft 8.1 system, open PeopleSoft Application Designer and open the *EXAMPLE_PSFT_MSG* message. Right-click *VERSION_1* and select Create test message. The Create Test Message window appears.

2. Expand Transaction in the tester window. Set the value for *EXAMPLE_CHAR.* Open the PSCAMA section and set the AUDIT_ACTN to *A* and click OK. A message is published. Wait a minute or two before proceeding to allow the message to be processed by both nodes.
3. On the PeopleSoft 8.1 system, open the Message Monitor to view the details of the outbound message.
4. On the PeopleSoft 8.50 system, open the Service Operation Monitor to view the details of the received message.

Example 4: Using the JMS Connectors

This section discusses using the JMS listening and JMS target connectors.

Understanding the JMS Connector Examples

The examples in this section are intended to be generic and independent of the JMS provider being used. Because of this, in certain steps general instructions are provided. The actual details of the task will depend on the provider being used – and may be rather involved. Please refer to the appropriate documentation.

The error queue is not configured in the examples. However, configuring the error queue may be desirable should issues arise while trying the examples.

The examples in this section focus on queues, but the process for using the JMS connectors when working with topics is essentially the same.

See Also

Chapter 6, "Using Listening Connectors and Target Connectors," Working With the JMS Connectors, page 90

Prerequisites

To use the examples in this section, a JMS provider must be configured and running. Please refer to the provider's documentation for instructions on how to accomplish these tasks. Ensure that messages can be sent to topics and queues before proceeding with the examples.

For the JMS target connector example, you will need a utility to consume and view the messages created. For the JMS listening connector example, you will need a utility to create the messages. The exact details of these utilities depend on the provider. Some may provide an administrative console that you can use to view the contents of topics and queues, and possibly send new messages to them. Other providers may include sample Java programs that you can use to interact with the provider. Refer to the provider's documentation for further details.

A special case exists for testing the JMS listening connector and queues when the provider is IBM MQSeries. In this instance, use Send Master to test the JMS listening connector.

See Also

Enterprise PeopleTools 8.50 PeopleBook: Integration Testing Utilities and Tools, "Using the Send Master Utility," Using JMS Projects

Using the JMS Target Connector

In this example, PeopleSoft Integration Broker will generate a JMS message, which will be consumed outside of the PeopleSoft system.

To use the JMS target connector:

1. On the JMS provider, create a JMS Connection Factory with the JNDI name *ExampleConnectionFactory*.
2. On the JMS provider, create a JMS Queue with the JNDI name *ExampleQueue.*
3. In PeopleSoft Application Designer, open the *EXAMPLE_WORKREC* record and add the following PeopleCode to the FieldChange event for the TEST field:

```
/* create an XML document */
&xmldata = "<?xml version='1.0'?><ConnectorTest/>";
&xmlDoc = CreateXmlDoc(&xmldata);
&rootNode = &xmlDoc.documentelement;

/* add text to the document */
&descNode = &rootNode.AddElement("TestNode");
&descNode.NodeValue = "Sending a message to a JMS queue.";

/* and send it out in an async request */

&MSG = CreateMessage(Operation.EXAMPLE_SERVICE_ASYNC_OPR);
&MSG.SetXmlDoc(&xmlDoc);
%IntBroker.Publish(&MSG);

MessageBox(0, "", 0, 0, "Message sent.");
```

4. In the PeopleSoft Pure Internet Architecture, open the node definition for *TARGETNODE.* Set the Connector ID to *JMSTARGET.* Set the values for the following properties:

Property	***Value***
JMSFactory	ExampleConnectionFactory.
JMSProvider	Name of the provider being used.
JMSUrl	Connection URL for the provider.
JMSQueue	ExampleQueue.
JMSUserName	The username on the JMS provider.
JMSPassword	The encrypted password for the user ID.

5. Test the connector:

   a. Open the test page, and click on the Test button.

   b. Verify that the message was sent to the queue. The exact mechanism for doing depends on the provider or utility that you are using.

Using the JMS Listening Connector

In this example, you will use the JMS listening connector to send a message to the JMS provider. PeopleSoft Integration Broker will consume the message.

To use the JMS listening connector:

1. On the JMS provider, create a JMS Connection Factory with the JNDI name *ExampleConnectionFactory*.

2. On the JMS provider, create a JMS Queue with the JNDI name *ExampleQueue.*

3. In PeopleSoft Application Designer, create a application package and application class. In the application class, put the following PeopleCode in the OnRequest function

```
Local XmlDoc &xmldoc;

&xmldoc = &_MSG.GetXmlDoc(); /*&_msg is the parameter*/
/* and write it to a file */
Local File &theFile = GetFile("JMSRequest.txt", "W");

&theFile.WriteString(&xmldoc.GenXmlString());
&theFile.Close(); /* create the reponse message */
Local Message &outmsg;
&outmsg = CreateMessage(Operation.EXAMPLE_SERVICE_OPR,
%IntBroker_Response);
/* build the body of the response */

Local string &xmldata = "<?xml version='1.0'?><ConnectorTest/>";
&xmldoc = CreateXmlDoc(&xmldata);

Local XmlNode &rootNode = &xmldoc.DocumentElement;

Local XmlNode &descNode = &rootNode.AddElement("ResponseMessage");
&descNode.NodeValue = "This wasgenerated in the OnRequest handler.";

/* add the body to the message */
&outmsg.SetXmlDoc(&xmldoc);

/* send the response message */
Return &outmsg;
```

4. In PIA, open the handler tab on the service operation EXAMPLE_SERVICE_OPR, and set the application class package, class and method name as you defined above.

5. In the integrationGateway.properties file, uncomment the following line:

```
ig.jms.Queues=1
```

6. Set the following properties to the values indicated:

Property	*Value*
ig.jms.Queue1	ExampleQueue
ig.jms.Queue1.Provider	<the name of the provider>
ig.jms.Queue1.JMSFactory	ExampleConnectionFactory
ig.jms.Queue1.Url	<connection URL for the provider>
ig.jms.Queue1.Use	< the userid >
ig.jms.Queue1.Password	<the encrypted password for the userid. >
ig.jms.Queue1.MessageName	EXAMPLE_SERVICE_OPR.VERSION_1
ig.jms.Queue1.RequestingNode	SOURCENODE
ig.jms.Queue1.DestinationNode	<the name of the local node >

7. Deploy and start the JMSListeningConnectorAdministrator servlet.

   See Chapter 6, "Using Listening Connectors and Target Connectors," Using the JMS Listening Connector, page 91.

8. Test the connector:

   a. Send a text message to the example JMS queue. Set the text of the message to:

```
<?xml version="1.0"?>
<ConnectorTest>
     <TestNode>Sending a message to the JMS Listening Connector.</TestNode>
</ConnectorTest>
```

   b. Check the message logs and the file named in the OnRequest method of application class . The message should be present in both.

Example 5: Using the AS2 Connectors

This section discusses using the AS2 listening and AS2 target connectors.

Understanding the AS2 Connector Examples

The purpose of the AS2 protocol is to allow the secure exchange of EDI data over the internet with trading partners. In the simplest case of an AS2 Message exchange, a sender packages data into an AS2 message structure and sends the message to trading partner over HTTP. Any kind of data can be transferred using AS2, including XML, EDI, text and binary.

This examples in this section demonstrate using the AS2 target connector to send an XML message to an external trading partner and using the AS2 listening connector to receive an XML message from a trading partner.

See Also

Chapter 6, "Using Listening Connectors and Target Connectors," Working With the AS2 Connectors, page 113

Prerequisites

To use the examples in this section, security certificates must be setup and registered in the keystore on the source and target machines. Take note of the certificate alias name for both the source or signer and the target or recipient servers, as you will need this information to set connector properties.

Verify that messages can be sent to and received from the AS2 external trading partner over HTTP before proceeding with the examples.

For the AS2 target connector example, you will need a third-party application to consume and view the messages created. For the AS2 listening connector example, you will need a third-party application to create and deliver the messages.

See Also

Chapter 11, "Setting Up Secure Integration Environments," page 155

Using the AS2 Target Connector

In this example PeopleSoft Integration Broker will generate an AS2 message and send it to a trading partner using HTTP. The external trading partner consumes the message. This example shows the tasks to perform to receive an MDN response message back synchronously or asynchronous.

To use the AS2 target connector:

1. In PeopleSoft Application Designer open the *EXAMPLE_WORKREC* record and add the following PeopleCode to the FieldChange event for the TEST field:

```
/*create an XML document */
Local string &xmldata;
Local XmlDoc &xmlDoc;
Local XmlNode &rootNode, &descNode;
Local boolean &result;

&xmldata = "<AS2ConnectorTest/>";

&xmlDoc = CreateXmlDoc("");
&rootNode = &xmlDoc.CreateDocumentElement("AS2ConnectorTest");
&rootNode = &xmlDoc.DocumentElement;

/* add text to the document */
&descNode = &rootNode.AddElement("TestNode");
&descNode.NodeValue = "Sending a AS2 message.";

&MSG = CreateMessage(Operation.EXAMPLE_SERVICE_ASYNC_OPR);
&MSG.SetXmlDoc(&xmlDoc);

/* and send it out in an async request */
%IntBroker.Publish(&MSG);

MessageBox(0, "", 0, 0, "AS2 Message sent.");
```

2. In the PeopleSoft Pure Internet Architecture, open the node definition for TARGETNODE. Set the Connector ID to *AS2TARGET.* Set the values for the following required properties:

Property Name	***Value***
URL	Specify the URL to which messages are sent using this connector.
AS2To	Specify the name of the sending node.
AS2From	Specify the name of the receiving node.
RecipientCertAlias	Specify the alias of the receiving certificate.
SignersCertificateAlias	Specify the alias of the signing certificate.

3. Add an outbound asynchronous transaction on the *AS2TARGETNODE,* to identify that the message *EXAMPLE_MESSAGE, VERSION_1* will be sent to the URL location.

4. Set the following properties in the integrationGateway.properties file. Use PSCipher.bat utility located at <PIA_HOME>\webserv\peoplesoft to encrypt the keystore password.

```
#AS2 Log Directory, logs all incoming and outgoing AS2 request and responses.
#Uncomment and specify the correct directory name to enable logging.
ig.AS2.LogDirectory = c://temp//as2

#AS2 Properties
#Uncomment the following two lines to specify your keystore and AS2 properties
ig.AS2.KeyStorePath=KeyStore Location (use // for windows path)
ig.AS2.KeyStorePassword=EncryptedKeyStorePassword
ig.AS2.AS2Directory=Location of AS2 Certificates  (required for Async MDN Type)
ig.AS2.LogDirectory=Path to store AS2 Log Files (optional)

Examples
ig.AS2.KeyStorePath=C://pt846-112-R2//webserv//peoplesoft//keystore//pskey
ig.AS2.KeyStorePassword=GD9klUFw8760HVaqeT4pkg==
ig.AS2.AS2Directory=c://temp//as2
ig.AS2.LogDirectory = c://temp//as2//logs
```

5. If the MDN response is synchronous, go to step 8.

   If the MDN response is asynchronous, verify the delivered node named *ASYNC_MDN* exists.

6. Verify that the node *ASYNC_MDN* has an active incoming asynchronous routing for the service operation *ASYNC_MDN_RESPONSE. VERSION_1*.

7. Verify that the delivered queue *AS2_CHANNEL* is not in *Pause* mode.

8. Test the connector.

   a. Open the test page, and click on the Test button.

   b. Verify that the message was sent to the recipient. The exact mechanism for doing so depends on the AS2 trading partner you are using.

   c. Verify that the MDN response was sent back to the source. The exact mechanism for doing so depends on the AS2 trading partner you are using.

9. If the MDN type is set to Async, verify that the ASYNC_MDN_RESPONSE was received.

Using the AS2 Listening Connector

In this example, you will use the AS2 listening connector to receive a message sent by the AS2 trading partner, and return an MDN synchronous or asynchronous response. Perform all tasks on the target machine. PeopleSoft Integration Broker will consume the message.

To use the AS2 listening connector:

1. In the PeopleSoft Pure Internet Architecture, choose the node that corresponds to the AS2 trading partner sending the message.

2. Insert an inbound asynchronous routing corresponding to the service operation *EXAMPLE_REQUEST_ASYNC_OPR.VERSION_1* expected.

3. Insert an outbound asynchronous routing corresponding to the service operation *EXAMPLE_RESPONSE_ASYNC_OPR.VERSION_1* as a reply.

4. In PeopleSoft Application Designer, create an application package and application class, and provide a method OnNotify with the following PeopleCode:

```
Local XmlDoc &xmlDoc;
   Local File &theFile;
   Local Message &msg;
  Local XmlDoc &MsgXmlDoc,  &xmlDoc;
   Local XmlNode &rootNode, &descNode;

   /* get the body of the incoming message */
   &MsgXmlDoc = &MSG.GetXmlDoc(); /* and write it to a file */
   &theFile = GetFile("AS2Request.txt", "W");
   &theFile.WriteString(&MsgXmlDoc.GenXmlString());
   &theFile.Close();

   &xmlDoc = CreateXmlDoc("");

   &rootNode = &xmlDoc.CreateDocumentElement("ConnectorTest");
   &rootNode = &xmlDoc.DocumentElement; /* add text to the document */

   &descNode = &rootNode.AddElement("ResponseMessage");
   &descNode.NodeValue = "This was generated in the OnRequest event.";
   /* send the response message */

   &msg = CreateMessage(Operation.EXAMPLE_RESPONSE_ASYNC_OPR);

   &msg.SetXmlDoc(&xmlDoc);

   /* and send it out in an async request */
   %IntBroker.Publish(&msg);
```

5. In PIA, open the handler tab on the service operation EXAMPLE_RESPONSE_ASYNC_OPR, and set the application package, class name and method.

6. In the integrationGateway.properties file, set the following properties to the values indicated:

```
#AS2 Properties
#Uncomment the following two lines to specify your keystore and AS2 properties
ig.AS2.KeyStorePath=KeyStore Location (use // for windows path)
ig.AS2.KeyStorePassword=EncryptedKeyStorePassword
ig.AS2.LogDirectory=Path to store AS2 Log Files (optional)

#example:
ig.AS2.KeyStorePath=C://pt846-112-R2//webserv//peoplesoft//keystore//pskey
ig.AS2.KeyStorePassword=GD9klUFw8760HVaqeT4pkg==
ig.AS2.LogDirectory = c://temp//as2//logs
```

In the following required properties, replace the <SOURCENODE> with the name of the AS2 trading partner source node, and <TARGETNODE> with the name of the local target node. Continue to set the value of the property.

```
# CertificateAlias is the certificate of AS2 Listening Node.
# SignerCertificateAlias is the certificate of AS2 trading partner of Listening⇒
 Node.
ig.AS2.QE_<SOURCE>.<TARGET>.CertificateAlias= Target Machine Alias
ig.AS2. <SOURCE>.<TARGET>.SignerCertificateAlias=Source Machine Alias

#example:
ig.AS2.PSFT_SRC_NODE.PSFT_TGT_NODE.CertificateAlias=<GeneratedAS2certificatealias>
ig.AS2.PSFT_SRC_NODE.PSFT_TGT_NODE.SignerCertificateAlias=<Generated⇒
AS2certificatealias>
```

The following values only need to be set if the incoming data does not contain the appropriate AS2To and AS2From values in the header of the message. It is best to leave these values in the request message header and leave these properties commented out.

```
#This map translate AS2From and AS2To to a different node name.
#This property is not required if you would use AS2FROM and AS2TO http header.
ig.AS2.AS2ListenerMap.From.<SOURCEALIAS> = Specify the Source Node Name
ig.AS2.AS2ListenerMap.To.<TARGETALIAS> = Specify the Target Node Name

#example:
ig.AS2.AS2ListenerMap.From.QE_SOURCE= PT_LOCAL
ig.AS2.AS2ListenerMap.To. QE_IBTGT= AS2TARGETNODE
```

7. Test the connector:

   a. Send a text message to the example AS2 queue. Name the message *EXAMPLE_REQUEST_MSG.*

   b. Set the text of the message to:

```
<?xml version="1.0"?>
<ConnectorTest>
<TestNode>Sending a message to the AS2 Listening Connector.</TestNode>
</ConnectorTest>
```

   c. Check the file named in the subscription PeopleCode. The default location for this file is <PS_CFG_HOME>\appserv\<DOMAIN_NAME>\Files. The message contents should be present.

   d. If the MDN type is asynchronous, verify that the AS2 trading partner received the *ASYNC_MDN_RESPONSE.*

Example 6: Using the Simple File Target Connector

This section describes how to use the simple file target connect to:

- Write PeopleSoft data to a file.
- Read data into PeopleSoft from a file.

Writing PeopleSoft Data to Files

This section describes how to use the simple file target connector to write PeopleSoft data to a file.

To write data to a file:

1. In PeopleSoft Application Designer, open the EXAMPLE_WORKREC record. Add the following PeopleCode to the FieldChange event for the TEST field:

```
&msg = CreateMessage(Operation.EXAMPLE_SERVICE_OPR);

&xmldata = "<?xml version='1.0'?><ConnectorTest/>";

/* create an XmlDoc */
&xmlDoc = CreateXmlDoc(&xmldata);
&rootNode = &xmlDoc.documentelement;
&descNode = &rootNode.AddElement("TestNode");
&descNode.NodeValue = "This message was written to a file.";

/* put the XML in the request... */
&msg.setxmldoc(&xmlDoc);

/* ...and send */
&response = %IntBroker.SyncRequest(&msg);
```

2. In the PeopleSoft Pure Internet Architecture, open the node definition TARGETNODE.
   a. On the Node Definitions-Connectors tab, set the Connector ID to FILEOUTPUT.
   b. Add the following connector properties and values:

Property	***Value***
Method	*PUT*
FilePath	Enter the location where you want the connector to write the file. For example, c:\temp.

3. Access the integrationGateway.properties file and comment out the following line.

   ```
   ig.fileconnector.password=EncryptedPassword
   ```

   The password option is not used in this example.

4. In the PeopleSoft Pure Internet Architecture, open the EXAMPLE_PAGE page and click Test.

5. Go to the directory specified in the connector properties, and open the new file. The contents should reflect what was created in the PeopleCode.

Reading Data Into PeopleSoft From Files

This section describes how to use the simple file target connector to read data into PeopleSoft from files.

To read data from files into PeopleSoft:

1. Create an XML file and include the following text:

```
<?xml version='1.0'?>
<ConnectorTest>
  <TestNode>
    The file has been read from the file system.
  </TestNode>
</ConnectorTest>
```

2. In PeopleSoft Application Designer, open the EXAMPLE_WORKREC record. Add the following PeopleCode to the FieldChange event for the TEST field:

```
&msg = CreateMessage(Operation.EXAMPLE_SERVICE_OPR);

&xmldata = "<?xml version='1.0'?><ConnectorTest/>";

/* create an XmlDoc */
&xmlDoc = CreateXmlDoc(&xmldata);

/* put the XML in the message */
&msg.setxmldoc(&xmlDoc);

/* send the request */
&response = %IntBroker.SyncRequest(&msg);

/* display the results */
&xmlDoc = &response.getxmldoc();
MessageBox(0, "", 0, 0, &xmlDoc.genxmlstring());
```

3. In the PeopleSoft Pure Internet Architecture, open the *TARGETNODE* node definition.

   a. Set the Connector ID to FILEOUTPUT.

      Set the FilePath property to the location from where the connector will read the output file.

   b. Add the following connector properties and values:

Property	***Value***
FileName	Specify the name of the output file. The file's default name has the following format: *sourcenodename.serviceoperationname.publication_id.xml*
METHOD	*GET*

4. Access the integrationGateway.properties file and comment out the following line.

   ```
   ig.fileconnector.password=EncryptedPassword
   ```

   The password option is not used in this example.

5. In the PeopleSoft Pure Internet Architecture, open the EXAMPLE_PAGE page and click the Test button.

   A message box appears that displays the data from the file read.

Example 7: Using the FTP Target Connector

This sections discusses how to use the FTP target connector to:

- Upload files to an FTP server.
- Download files from an FTP server.

Prerequisites

For the examples presented in this section, you must have an active FTP server, as well as an account on that server.

Uploading Files to FTP Servers

To upload a file to an FTP server:

1. In PeopleSoft Application Designer, open the EXAMPLE_WORKREC record and add the following PeopleCode to the FieldChange event for the TEST field:

```
&msg = CreateMessage(Operation.EXAMPLE_SERVICE_OPR);

&xmldata = "<?xml version='1.0'?><ConnectorTest/>";

/* create an XmlDoc */
&xmlDoc = CreateXmlDoc(&xmldata);
&rootNode = &xmlDoc.documentelement;
&descNode = &rootNode.addelement("FTPtest");
&descNode.nodevalue = "This text will be uploaded";

/* put the XML in the message */
&msg.setxmldoc(&xmlDoc);

/* send the request */
&response = %IntBroker.SyncRequest(&msg);
```


2. In the PeopleSoft Pure Internet Architecture, open the *TARGETNODE* node definition.

   a. On the Node Definitions-Connectors tab, set the Connector ID to *FTPTARGET*.

   b. Set the following properties to the values indicated:

Property	***Value***
HOSTNAME	Specify the IP address or name of the FTP server for the connection.
METHOD	*PUT*
USERNAME	Enter the FTP server login ID.
PASSWORD	Enter the password for the login to the FTP server. This password must be encrypted. Use the Password Encryption Utility at the bottom of the page to encrypt the password, if necessary

3. In the PeopleSoft Pure Internet Architecture, open the EXAMPLE_PAGE page and click the Test button.

   Login to the FTP server and check for the file. Open the file and verify the contents.

Downloading Files From FTP Servers

To download a file from an FTP server:

1. Create an XML file with the following contents and place the file on an FTP server.

```
<?xml version="1.0"?>
<ConnectorTest>
<TestNode>This message will be downloaded from an FTP server.</TestNode>
</ConnectorTest>
```

2. In PeopleSoft Application Designer, open the EXAMPLE_WORKREC record and add the following PeopleCode to the FieldChange event for the TEST field:

```
&msg = CreateMessage(Operation.EXAMPLE_SERVICE_OPR);

&xmldata = "<?xml version='1.0'?><ConnectorTest/>";

/* create an XmlDoc */
&xmlDoc = CreateXmlDoc(&xmldata);

/* put the XML in the message */
&msg.setxmldoc(&xmlDoc);

/* send the request */
&response = %IntBroker.SyncRequest(&msg);

/* display the contents */
&xmlDoc = &response.getxmldoc();
MessageBox(0, "", 0, 0, &xmlDoc.genxmlstring());
```

3. In the PeopleSoft Pure Internet Architecture, open the *TARGETNODE* node definition.

   a. On the Node Definitions-Connectors tab, set the Connector ID to *FTPTARGET*.

   b. Set the following properties to the values indicated:

Property	***Value***
HOSTNAME	Specify the IP address or name of the FTP server for the connection.
METHOD	*GET*
FILENAME	Specify the name of the file.
USERNAME	Enter the FTP server login ID.
PASSWORD	Enter the password for the login to the FTP server. This password must be encrypted. Use the Password Encryption Utility at the bottom of the page to encrypt the password, if necessary

4. In the PeopleSoft Pure Internet Architecture, open the EXAMPLE_PAGE page and click the Test button.

   The contents of the XML file will display in the message box.

Example 8: Using the SMTP Target Connector

This section provides an example of how to use the Simple Mail Transfer Protocol (SMTP) target connector to send an email message using an SMTP server.

Prerequisites

For this example, you must have an active SMTP server as well as an active email account to receive the message.

Sending Email Messages to SMTP Servers

To send an email message to an SMTP server using the SMTP target connector:

1. In PeopleSoft Application Designer, open the EXAMPLE_WORKREC record and add the following PeopleCode to the FieldChange event for the TEST field:

```
&msg = CreateMessage(Operation.EXAMPLE_SERVICE_OPR);

&xmldata = "<?xml version='1.0'?><ConnectorTest/>";

/* create an XmlDoc */
&xmlDoc = CreateXmlDoc(&xmldata);
&rootNode = &xmlDoc.documentelement;
&descNode = &rootNode.addelement("SMTPtest");
&descNode.nodevalue = "This xml will appear in the email";

/* put the XML in the message */
&msg.setxmldoc(&xmlDoc);

/* send the request */
&response = %IntBroker.SyncRequest(&msg);
```

2. In the PeopleSoft Pure Internet Architecture, open the *TARGETNODE* node definition.

   a. On the Node Definitions-Connectors tab, set the Connector ID to *SMTPTARGET*.

   b. Set the following properties to the values indicated:

Property	***Value***
DestEmailAddress	Set this property to the email address to which the email will be sent.
SourceEmailAddress	Set this property to the email address from which you are sending the message.

3. Access the integrationGateway.properties file. Locate the following line and replace <mailServerName> with the name of the SMTP server.

```
ig.connector.smtptargetconnector.host=<mailServerName>
```

4. In the PeopleSoft Pure Internet Architecture, open the EXAMPLE_PAGE page and click the Test button to send the message.

   Check the destination email account for the message. Since the message is being passed through one or more SMTP servers, there may be some propagation delay and the message might not be received immediately.

Appendix B

Using the Integration Broker Connector SDK

This chapter provides an overview of the PeopleSoft Integration Broker connector software development kit (SDK) and discusses how to:

- Develop connectors.
- Develop connectors based on the document object model (DOM).
- Develop and implement connectors in the C/C++ environment.
- Reuse connector code.

Understanding the PeopleSoft Integration Broker Connector SDK

This section discusses:

- The PeopleSoft Integration Broker Connector SDK.
- SDK contents.
- SDK location.
- SDK application programming interface (API) documentation.

The PeopleSoft Integration Broker Connector SDK

Target connectors generate message requests, send them to integration participants, wait for responses from participants, and deliver the responses back to the gateway manager. Listening connectors receive message requests from integration participants, send them to the gateway manager, and deliver responses back to the integration participants.

PeopleSoft Integration Broker is bundled with connectors for use with PeopleSoft, HTTP, Java Messaging Service (JMS), PeopleSoft 8.1x, File Transfer Protocol (FTP), and Simple Mail Transfer Protocol (SMTP) communication formats. You can use the PeopleSoft Integration Broker Connector SDK to build and implement connectors for other communication formats and application requirements.

For example, you could develop a connector for integrations between a PeopleSoft Human Resources system and an SAP Financials system. You could also develop a connector for integrations between a PeopleSoft Supply Chain system and a supplier by using a RosettaNet system or for integrations between a PeopleSoft eProcurement system and a Commerce One Marketsite by using an XML Common Business Library (XCBL) or SAdt eXplain (SAX) system.

SDK Contents

The PeopleSoft Integration Broker Connector SDK includes:

- Instructions for setting up the development environment.
- Java classes that are required for creating connectors (including IBResponse and IBRequest objects).
- API documentation.
- Sample code for listening and target connector classes.
- A Send Master utility to test connectors.
- A Simple Post utility that enables third-party systems to post messages to the integration gateway.

SDK Location

The following table lists the location of the SDK and its contents.

Item	*Location*
SDK	<PIA_HOME>\webserv\<DOMAIN>\applications\peoplesoft\PSIGW.war\SDK
Java classes	<PIA_HOME>\webserv\<DOMAIN>\applications\peoplesoft\PSIGW.war\WEB-INF\classes
API documentation	<PIA_HOME>\webserv\<DOMAIN>\applications\peoplesoft\PSIGW.war\SDK\docs\index.html
Sample code for listening and target connector classes	<PIA_HOME>\webserv\<DOMAIN>\applications\peoplesoft\PSIGW.war\SDK\src
Send Master utility	<PIA_HOME>\webserv\<DOMAIN>\StartSendMaster.bat, or <PIA_HOME>\webserv\<DOMAIN> \StartSendMaster.sh
Simple Post utility	<PIA_HOME>\webserv\<DOMAIN>\applications\peoplesoft\PSIGW.war\WEB-INF\classes\com\peoplesoft \pt\simplepost

SDK API Documentation

The PeopleSoft Integration Broker Connector SDK includes API documentation in HTML format.

The documentation lists and describes Java packages that you can use to develop custom connectors and includes information about package classes, inner classes, interfaces, constructors, methods, and fields.

Access the Connector API documentation in the connector SDK directory, <PIA_HOME>\webserv\<DOMAIN>\applications\peoplesoft\PSIGW.war\SDK\docs\index.html.

Developing Connectors

This section provides overviews of connector development and implementation and general connector class development considerations and discusses how to:

- Develop target connector classes.
- Develop listening connector classes.
- Install connector classes.
- Register connectors.
- Use connector templates.

Understanding Connector Development and Implementation

You can produce new connectors in different ways, based on whether you want to create a listening connector or a target connector.

Listening connectors use standard connector interface and gateway services to link to the integration gateway. Although a Java interface object is not used for listening connectors, the listening connectors still must adhere to a standard scheme of logic to drive requests to, and to process responses from, the integration gateway.

Target connectors must implement a Java interface to become valid target connectors in the integration gateway. This ensures a standard interface for the gateway manager so that it can manage each target connector in a streamlined way.

To develop connectors, you:

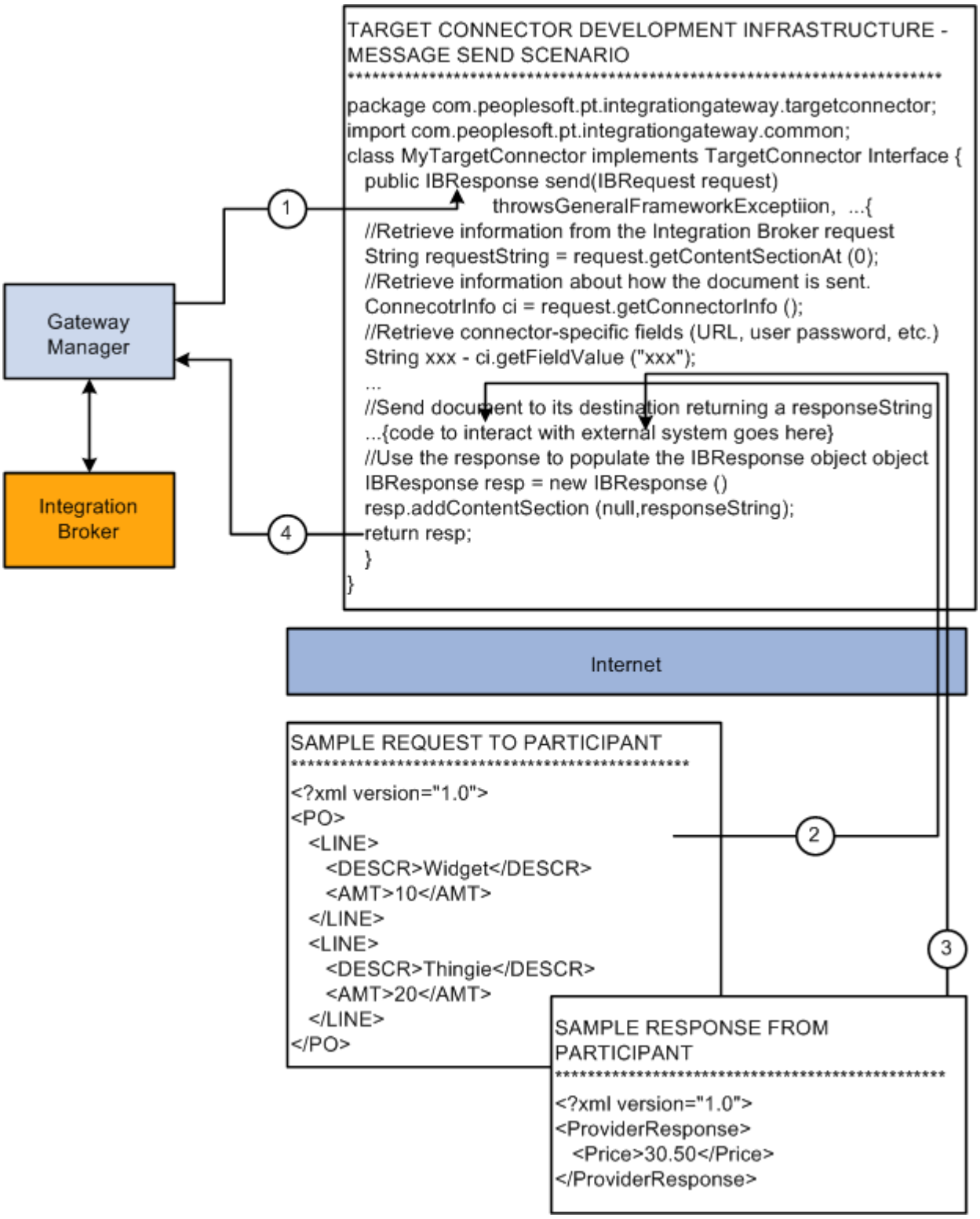
1. Develop a connector class.
2. Install the connector class.
3. Register the connector (target connectors only).

Target and listening connector templates are provided at the end of this chapter for you to use as a starting point for connector development.

Following are various scenarios for the target connector development infrastructure.

Message Send Scenario for the Target Connector Development Infrastructure

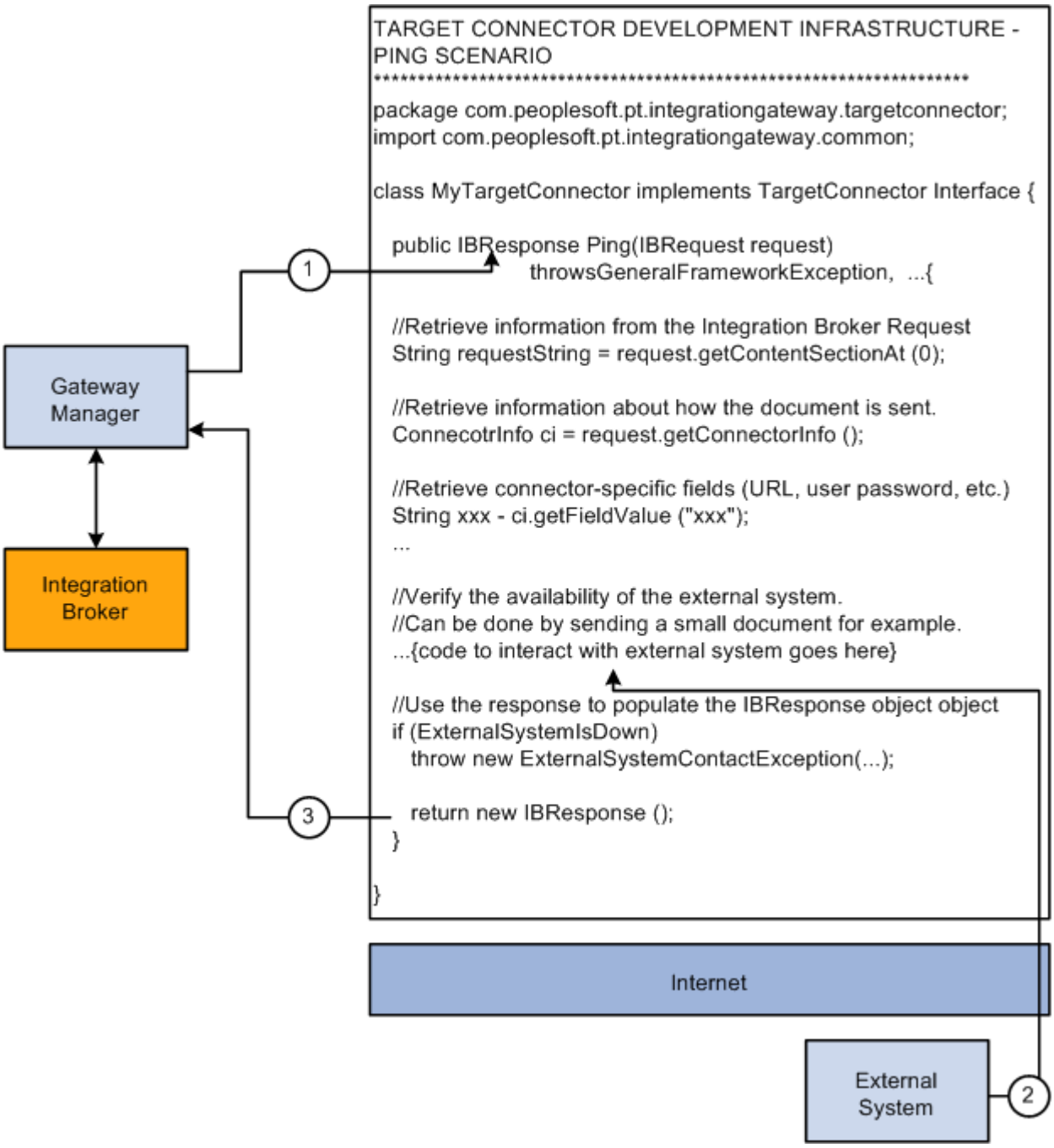
You develop target connectors to generate and send message requests to integration participants and return responses. This diagram shows how the connector code accomplishes these tasks:



Message send scenario

Ping Scenario for the Target Connector Development Infrastructure

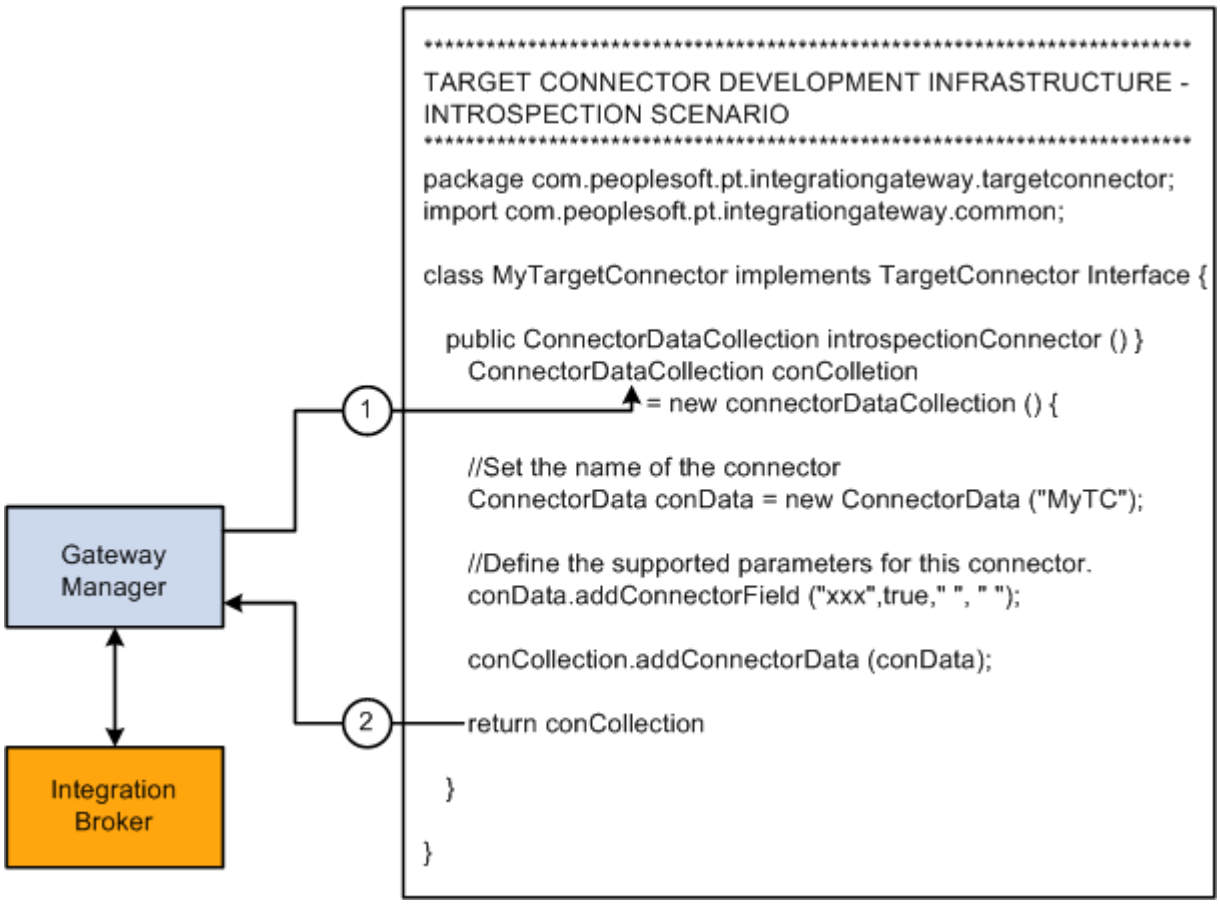
You can develop functionality in target connectors to ping external systems. This diagram shows how the connector code accomplishes this task:



Ping scenario

Introspection Scenario for the Target Connector Development Infrastructure

You can develop functionality within target connectors to handle connector introspection. This diagram shows how the connector code accomplishes this task:



Introspection scenario

Understanding General Connector Class Development Considerations

While implementations vary greatly, when you develop connector classes, you should incorporate specific functionality.

Input and Output Formats That Are Exchanged Through Connectors

For a target connector to handle input and output formats that are exchanged with its intended recipient, it must transform the PeopleSoft Integration Broker request (IBRequest) into a message that is formatted for the intended external system.

For instance, the HTTP target connector that is delivered with PeopleSoft Integration Broker gathers HTTP headers and cookies from the IBRequest and formulates the appropriate HTTP message, complete with the actual message content, so that it can be delivered to its destination. When the response comes back, the connector creates a PeopleSoft Integration Broker response (IBResponse) by using the response string.

For a listening connector to handle input and output formats that are exchanged with its requestor, it must transform the incoming message into an IBRequest. For example, the HTTP listening connector that is delivered with PeopleSoft Integration Broker recognizes SOAP messages and retrieves query string arguments, HTTP headers, and cookies. It then formats all of this information to create the IBRequest so that PeopleSoft Integration Broker can converse with it. When the response comes back, the HTTP listening connector reads the IBResponse and sends its output message content back to the requesting system.

Interaction Between Local and External Systems

A target connector interacts with an external system by sending it information and by retrieving the response.

For example, to accomplish this interaction, the HTTP target connector that is delivered with PeopleSoft Integration Broker uses various HTTP-specific classes to send messages through HTTP and to handle the external system being down, security (through HTTPS), and so forth.

A listening connector interacts with an external system by receiving requests from the external system and returning responses that the external system understands. For example, to accomplish this interaction, the HTTP listening connector that is delivered with PeopleSoft Integration Broker uses a servlet to receive and reply to incoming HTTP messages.

Developing Target Connector Classes

This section discusses target connector class development issues to consider when creating custom target connectors.

Using the Target Connector Interface

As with PeopleSoft-provided target connectors, the integration gateway dynamically invokes custom target connectors through the gateway manager.

Target connectors must adhere to a standard structure as defined in the target connector interface.

```
public interface TargetConnector {
   IBResponse send(IBRequest request) throws
                    GeneralFrameworkException,
                    DuplicateMessageException,
                    InvalidMessageException,
                    ExternalSystemContactException,
                    ExternalApplicationException,
                    MessageMarshallingException,
                    MessageUnmarshallingException;

   IBResponse ping(IBRequest request) throws
                    GeneralFrameworkException,
                    DuplicateMessageException,
                    InvalidMessageException,
                    ExternalSystemContactException,
                    ExternalApplicationException,
                    MessageMarshallingException,
                    MessageUnmarshallingException;

ConnectorDataCollection introspectConnector();
```

Use the Send method to send a request to an external system and to retrieve its response. The gateway manager passes the request to this method and expects a response to be returned.

The Ping method enables PeopleSoft Integration Broker to verify the availability of a site. The Integration Broker Monitor can also invoke the Ping method explicitly.

ConnectorDataCollection invokes introspection.

Use gateway services, such as XMLDoc, for document access and mutation. You also have access to the IBResponse and IBRequest objects from the integration gateway.

Building Introspection into Target Connectors

PeopleSoft Integration Broker can introspect (query) the capabilities of target connectors that are installed on a local or remote integration gateway by using introspection. Load all target connectors that are delivered with PeopleSoft Integration Broker by clicking the Load button on the Connectors page in the Gateways component.

You can build introspection into custom-built connectors. When you do so, you can load the connector and its properties with the click of a button.

For the introspection process to gather information about a custom target connector, you must implement the IntrospectConnector method.

The following example from the SMTP target connector sends messages through email:

```
public ConnectorDataCollection introspectConnector() {
   //Creates the ConnectorDataCollection that will be returned
   //by this method. This object will contain all the
   //necessary information about this Connector's properties.

      ConnectorDataCollection conCollection = new ConnectorDataCollection();

      //Create ConnectorData Object and stipulating the name of
      //the connector as seen from the Gateway Component.

      ConnectorData conData = new ConnectorData("SMTPTARGET");

      conData.addConnectorField("DestEmailAddress", true, "", "");
      conData.addConnectorField("SourceEmailAddress", true, "", "");
      conData.addConnectorField("CC", false, "", "");
      conData.addConnectorField("BCC", false, "", "");
      conData.addConnectorField("HEADER", Content-type", false,
         "", "text/plain|text/html");
      conData.addConnectorField("HEADER","sendUncompressed",true,
         "Y","Y|N");

      //Add the ConnectorData to your ConnectorDataCollection
      //Object. Typically, you would only
      //add one ConnectorData into your ConnectorDataCollection.
      conCollection.addConnectorData(conData);

   return conCollection;
}
```

Use the addConnectorField method to add connector fields:

addConnectorField *([PropertyID] PropertyName, Required, DefaultValue, Possible⇒ Values)*

Use the following parameters for this method:

Parameter	*Description*
Property ID	Identifies different property types, such as HEADER for HTTP or SMTP. PeopleSoft software also uses the HEADER property ID to allow a message to be sent in either compressed or clear format through the sendUncompressed property. If this parameter is not supplied, the property ID is the connector name.

Parameter	*Description*
Property Name	Identifies the name of the connector property.
Required	Determines whether the information is required for the target connector to deliver its message. Values are: • *Y:* True • *N:*False
Default Value	Identifies the default value for the property. Typically, you set the Required parameter to True when you specify a default value so that this information carries to the node configuration in the integration engine.
Possible Values	Identifies a list of the possible values that the property can take, separated by the | character.

The following definition shows how these properties function:

```
conData.addConnectorField("HEADER","sendUncompressed",true,"Y, "Y|N");
```

In this case, the property name is sendUncompressed and its property ID is HEADER. The sendUncompressed property is required (the third parameter is set to true), so that whenever you create a node in the node definition component and specify SMTPTARGET as the target connector, this property appears on the page automatically. Further, because the default value is set to *Y*, this is the default value. Possible values have been identified as *Y* or *N*. If you click the list box (search box) for this property on the Connectors tab in the Node Definition component, you can select from those two values.

Building Error Handling and Logging into Target Connectors

The following code example demonstrates how to build error handling and logging into target connectors:

```
package com.peoplesoft.pt.integrationgateway.targetconnector;

import ...

public class SampleTargetConnector implements TargetConnector {
         public IBResponse ping(IBRequest request)

         public IBResponse send(IBRequest request)throws
                     GeneralFrameworkException,
                     InvalidMessageException,
                     ExternalSystemContactException,
                     ExternalApplicationException,
                     MessageMarshallingException,
                     MessageUnmarshallingException,
                     DuplicateMessageException  {

      PSHttp httpObj = null;
      try {

      // Get handle on rootnode
      XmlNode root = dom.GetDocumentElement();

      // Cast the IBRequest back to an InternalIBRequest
      InternalIBRequest request = (InternalIBRequest)requestOrig;

      // Populate App Msg XML  Dom Object from IBRequest

      ...

      // Get the URL from either the IBRequest or from the
      //prop file (default)

      String URL = request.getConnectorInfo().getFieldValue("URL");

      // Log the request
      Logger.logMessage("SampleTargetConnector:
      Application Message Request", dom.GenerateXmlString(),
      Logger.STANDARD_INFORMATION);

      // Send the request DOM Document
      httpObj.setRequestProperty("content-type", "text/plain");
      httpObj.send(dom.GenerateXmlString().getBytes());

      // Get the response and convert to a String
      responseString = new String(httpObj.getContent());

      // Log the response
      Logger.logMessage("SampleTargetConnector:
      Application Message Response", responseString,
      Logger.STANDARD_INFORMATION);

      // Construct the IBResponse
      response = new IBResponse();

      ...

      // Return the successful IBResponse
      return response;

   } catch (XmlException xe) {
      httpObj.disconnect();
      throw new GeneralFrameworkException ("SampleTargetConnector:Failed
         while parsing XML");
```



```
   } catch (org.w3c.www.protocol.http.HttpException httpe) {
      throw new ("SampleTargetConnector:HTTP Protocol
         exception",httpe);

   } catch (java.io.IOException ioe) {
      throw new ExternalSystemContactException
         ("SampleTargetConnector:I/O Exception",ioe);

   } finally {
      httpObj.disconnect();
   }
   } // end send()
}
```

Developing Listening Connector Classes

This section discusses listening connector class development issues.

Building Servlet-Based and Nonservlet-Based Listening Connectors

If you require a listening connector that services HTTP requests, build a servlet-based listening connector. A servlet-based listening connector runs in the Servlet container on the web server.

See Appendix B, "Using the Integration Broker Connector SDK," Using Connector Templates, page 316.

This PeopleBook does not discuss how to install servlets on web servers.

See The servlet documentation for your web server.

Invoking Listening Connectors

Listening connectors must invoke PeopleSoft Integration Broker through the gateway manager Connect method.

```
IBResponse connect(IBRequest) throws
   GeneralFrameworkException
   DuplicateMessageException
   InvalidMessageException
   MessageMarshallingException
   MessageUnmarshallingException
   ExternalSystemContactException
   ExternalApplicationException
```

Controlling Message Routing

By accessing and modifying key information on the IBRequest, you can control the behavior of transactions as they flow through the integration gateway.

This section describes several dispatching features that you can use to control message routing by modifying the IBRequest from the listening connector, including routing messages to:

- Other (remote) integration gateways.
- Specific target connectors.
- Other PeopleSoft systems.

You can control the routing of a message to another integration gateway by specifying the uniform resource locator (URL) of the gateway in the IBRequest. You might need to forward messages to another gateway so that they can be processed by a remote PeopleSoft Integration Broker system. To do so, specify the URL of this integration gateway as follows:

```
. . .
IBRequest ibRequest = new IBRequest();
IbRequest.setOperationName("RemoteRoutingTest");
IbRequest.setRequestingNode("SourceSystem");
IbRequest.setPassword("myPassword");
. . .

//Specify the processing of the message to occur from
//anotherIntegration Gateway.
ibRequest.setRemoteFrameworkURL("https://hostName/PSIGW/
PeopleSoftListeningConnector");
```

You can also route a message to a specific target connector by modifying the request's ConnectorInfo object as follows:

```
. . .
IBRequest ibRequest = new IBRequest();
. . .

// Send a message through the HttpTargetConnector for example.
ConnectorInfo connectorInfo = ibRequest.getConnectorInfo();

connectorInfo.setConnectorClassName("HttpTargetConnector");
connectorInfo.setField("URL","http://www.externalsite.com");
connectorInfo.setField("Method","POST");
. . .
```

You can control the routing of messages to PeopleSoft Integration Broker systems by setting the destination node for messages and by configuring the integrationGateway.properties file. For example, suppose that you currently have PeopleSoft Human Resources, Financials, and Customer Relationship Management (PeopleSoft CRM) systems installed, and you have built a listening connector to listen to events on an SAP MRP system. A method called getDestinationSystem() on the listening connector lets you know the destination of each message. To call getDestinationSystem(), set the application server settings in the integrationGateway.properties file as follows:

```
. . .
//Default PeopleSoft System
ig.isc.serverURL=//HRSERVER:9000
ig.isc.userid=HRUSERID
ig.isc.password=HRPASSWORD
ig.isc.toolsRel=8.50

#
## JOLT connect string settings for Application Server(s) with known NODENAMEs.
#
ig.isc.FINANCIALS.serverURL=//FINSERVER:9000
ig.isc.FINANCIALS.userid=FINUSERID
ig.isc.FINANCIALS.password=FINPASSWORD
ig.isc.FINANCIALS.toolsRel=8.50

ig.isc.CRM.serverURL=//CRMSERVER:9000
ig.isc.CRM.userid=CRMUSERID
ig.isc.CRM.password=CRMPASSWORD
ig.isc.CRM.toolsRel=8.50
```

Make the following modifications to the listening connector:


```
. . .
IBRequest ibRequest = new IBRequest();
IbRequest.setOperationName("RoutingTest");
IbRequest.setRequestingNode("SourceSystem");
IbRequest.setPassword("myPassword");
. . .

// Get the name of the DestinationSystem from your proprietary
//method. This method would return one of the following
//("HR", //"FINANCIALS", "CRM").
String destinationSystem = getDestinationSystem();
ibRequest.setDestinationNode(destinationSystem);
. . .
// Create a GatewayManager instance.
GatewayManager gm = new GatewayManager();

// Invoke the Connector Framework Manager.
ibResponse = gm.connect(ibRequest);
. . .
```

Building Error Handling and Logging into Listening Connectors

This is sample code for building error handling and logging into listening connectors:

```
package com.peoplesoft.pt.integrationgateway.listeningconnector;

import ...

public class HttpListeningConnector extends HttpServlet {
      public void doGet(HttpServletRequest req,
      HttpServletResponse resp) throws ServletException, IOException {
      }

      public void doPost(HttpServletRequest req,
      HttpServletResponse resp) throws ServletException, IOException {

         String actualResponse ="";
         IBRequest request = null;
         IBResponse response = null;

      try {

         String inputString = MiscUtil.readerToString(new
         InputStreamReader(req.getInputStream()));

         // Log the actual Input String
         Logger.logMessage("HttpListeningConnector: HTTP
         Request", inputString, Logger.STANDARD_INFORMATION);

         HttpListeningConnectorUtility util = new
         HttpListeningConnectorUtility();
         request = util.createIBRequest("XML", req, inputString);

         // Use the GatewayManager to invoke the Integration
         // Server and return its response.
         GatewayManager conMgr = new GatewayManager();
         response = conMgr.connect(request);

         // Need to get the actual response from the
         //IBResponse
         actualResponse = response.getContentSectionAt(0);

   } catch (InvalidMessageException ime) {
         ime.printStackTrace();
         actualResponse = getErrorXml(ime);
         Logger.logError("HTTPListeningConnector:
         InvalidMessageException", request, response,
         Logger.STANDARD_GATEWAY_EXCEPTION, ime);

   } catch (ExternalSystemContactException esce) {
         esce.printStackTrace();
         actualResponse = getErrorXml(esce);
         Logger.logError("HTTPListeningConnector:
         ExternalSystemContactException", request, response,
         Logger.STANDARD_GATEWAY_EXCEPTION, esce);

   } catch (ExternalApplicationException esee) {
      esee.printStackTrace();
      actualResponse = getErrorXml(esee);
      Logger.logError("HTTPListeningConnector:
      ExternalApplicationException", request, response,
      Logger.STANDARD_GATEWAY_EXCEPTION, esee);

   } catch (MessageMarshallingException mme) {
      mme.printStackTrace();
      actualResponse = getErrorXml(mme);
      Logger.logError("HTTPListeningConnector:
      MessageMarshallingException", request, response,
```



```
            Logger.STANDARD_GATEWAY_EXCEPTION, mme);

         } catch (MessageUnmarshallingException mue) {
            mue.printStackTrace();
            actualResponse = getErrorXml(mue);
            Logger.logError("HTTPListeningConnector:
            MessageUnmarshallingException", request, response,
            Logger.STANDARD_GATEWAY_EXCEPTION, mue);

         } catch (GeneralFrameworkException gfe)  {
            gfe.printStackTrace();
            actualResponse = getErrorXml(gfe);
            Logger.logError("HTTPListeningConnector:
            GeneralFrameworkException", request, response,
            Logger.STANDARD_GATEWAY_EXCEPTION, gfe);
         }

         // Return the message to the original requestor that
         //invoked the Servlet
         HttpListeningConnectorUtility.
               sendResponseBackToRequestor(actualResponse, resp);

         // Log the actual output String
         Logger.logMessage("HttpListeningConnector:
            HTTP Response", actualResponse, Logger.STANDARD_INFORMATION);

         }   // end doPost()
      }
```

Installing Connector Classes

Install connector classes on the local web server.

Target Connector Classes

To install a target connector class, copy the class from the Java Classes directory to the following location on the local web server:

<PIA_HOME>\webserv\<DOMAIN>\applications\peoplesoft\PSIGW.war\WEB-INF\classes\com\peoplesoft\pt\integrationgateway\targetconnector

Listening Connector Classes

To install a listening connector class, copy the class to the following location on the local web server:

<PIA_HOME>\webserv\<DOMAIN>\applications\peoplesoft\PSIGW.war\WEB-INF\classes\com\peoplesoft\pt\integrationgateway\listeningconnector

Registering Connectors

Before you can use a target connector, you must register it on the integration engine. To register a connector, load the connector information in the Gateways component by using the Load button. Loading the connector makes its capabilities known to PeopleSoft Integration Broker.

Then, assign the connector to the intended node on the Connector page in the Node Definition component. Enter the connector ID that corresponds to the new connector and edit the properties, as needed.

See Also

Chapter 5, "Managing Integration Gateways," Loading Target Connectors, page 35

Using Connector Templates

This section contains the following generic connector templates to use as a starting point for connector development:

- Target connector template.
- Listening connector template (servlet).
- Listening connector template (nonservlet).

Target Connector Template

Use the following example as a target connector template:


```
package com.peoplesoft.pt.integrationgateway.targetconnector;
import com.peoplesoft.pt.integrationgateway.common;

class  MyTargetConnector implements TargetConnector Interface  {

   public IBResponse send(IBRequest request) throws
                  GeneralFrameworkException
                  DuplicateMessageException
                  InvalidMessageException
                  ExternalSystemContactException
                  ExternalApplicationException
                  MessageMarshallingException
                  MessageUnmarshallingException  {

      //Retrieve information from the Integration Broker
      //Request.
      String requestString = request.getContentSectionAt(0);

      //Retrieve Information about how the document is sent.
      ConnectorInfo ci = request.getConnectorInfo();

      //Retrieve Connector specific fields (URL, user, password
      //for example).
      String xxx = ci.getFieldValue("xxx");
      ...

      // Send document to its destination returning a
      //responseString.
      ...{code to interact with external system goes here}

      // Use the response to populate the ISResponse object
      IBResponse resp = new IBResponse()
      resp.addContentSection(null,responseString);
      return resp;
   }

   public IBResponse ping(IBRequest request)  throws
                  GeneralFrameworkException
                  DuplicateMessageException
                  InvalidMessageException
                  ExternalSystemContactException
                  ExternalApplicationException
                  MessageMarshallingException
                  MessageUnmarshallingException   {

      //Retrieve Information about how the document is sent.
      ConnectorInfo ci = request.getConnectorInfo();

      //Retrieve Connector specific fields (URL, user, password
      //for example).
      String xxx = ci.getFieldValue("xxx");
      ...

      // Send a simple document to its destination just to see if
      //the destination is up.
      ...{code to interact with external system goes here. Throw
         exceptions as needed.}

      // Return an empty IBResponse object
      return new IBResponse();
   }

   public ConnectorDataCollection introspectConnector() {
   ConnectorDataCollection conCollection = new
```

```
   ConnectorDataCollection();

      // Set the name of the connector.
      ConnectorData conData = new ConnectorData("MyTC");

      // Define the supported parameters for this Connector.
      conData.addConnectorField("xxx",true,"","");
      ...

      conCollection.addConnectorData(conData);

      return conCollection;
   }
}
```

Listening Connector Template (Servlet)

Use the following example as a template for a servlet-based listening connector:


```
package com.peoplesoft.pt.integrationgateway.listeningconnector;
import com.peoplesoft.pt.integrationgateway.common;

//This is an example of using a Servlet when receiving the //External Request.
public class MyListeningConnector extends HttpServlet {

   public void service (HttpServletRequest request,
   HttpServletResponse response)
            throws ServletException, IOException {

   //Developer retrieves the request and gets key information
   //here (such as UserName, password, operationName and
   //messageContent)
   ...

   //Developer creates the IBRequest object
   IBRequest request = new IBRequest();

   //Required information for an Integration Broker Request.
   request.setRequestingNode(UserName);
   request.setPassword(password);
   request.setOperationName(integrationService);
   request.addContentSection(null,messageContent);

   // Keep populating the IBRequest as needed (other set
   //methods are available)

   GatewayManager gatewayManager = new GatewayManager();
   try {
      //Send the request into the Integration Broker.
      IBResponse response = gatewayManager.connect(request);

      //Hand back your response to the requestor.
      out.println(response.getContentSectionAt(0));
   } catch(MashallingException me) {

      // Handle Marshalling errors here. For example :
      out.println("<?xml version=\"1.0\"?>"+
            "<MyResponse>"+
               "<Status>"+
               "<Code>1001</Code>"+
               "<Description>
               MarshallingException Occurred
               </Description>"+
               "</Status>"+
            "</MyResponse>");
   } catch(UnmarshallingException ume) {
      // Handle UnmarshallingException  here.
   } catch(. . . ) {
      // Handle all other Integration Broker Exceptions.
   } finally  {
      //Cleanup work here.  For example:
      out.close();
   }
}
```

Listening Connector Template (Non-servlet)

Use the following example as a template for non-servlet-based listening connector:

```
package com.peoplesoft.pt.integrationgateway.listeningconnector;
import com.peoplesoft.pt.integrationgateway.common;

//This is an example of a regular class used to receive the //External Request.
public class MyListeningConnector {

   //Somehow the external system makes a method call to this
   //method. Of course the Class and Method Name is entirely up
   //to the developer.

   public String callMyListeningConnector (String Request) {

   //Developer retrieves the request and gets key information
   //here (such as UserName, password, operationName and
   //messageContent)

   //Developer creates the IBRequest object
   IBRequest request = new IBRequest();

   //Required information for an Integration Broker Request.
   request.setRequestingNode(UserName);
   request.setPassword(password);
   request.setOperationName(integrationService);
   request.addContentSection(null,messageContent);

   //Keep populating the IBRequest as needed (other set
   //methods are available)

   GatewayManager gatewayManager = new GatewayManager();
   try {
   //Send the request into the Integration Broker.
   IBResponse response = gatewayManager.connect(request);

   //Hand back your response to the requestor.
   return response.getContentSectionAt(0);
   } catch(MashallingException me) {

      // Handle Marshalling errors here. For example :
      return ("<?xml version=\"1.0\"?>"+
         "<MyResponse>"+
            "<Status>"+
               "<Code>1001</Code>"+
               "<Description>
               MarshallingException Occurred
               </Description>"+
            "</Status>"+
         "</MyResponse>");
   } catch(UnmarshallingException ume) {

      // Handle UnmarshallingException  here.
   } catch(. . . ) {

      // Handle all other Integration Broker Exceptions.
   } finally  {
      //Cleanup work here (if any). . .
   }
}
```


Developing Connectors Based on DOM

PeopleSoft provides a Java XML DOM wrapper that enables you to manipulate message objects with PeopleCode instead of with a standard XML parser.

The section provides an overview of the Java XML DOM wrapper and discusses how to use Java XML DOM wrapper classes.

Understanding the Java XML DOM Wrapper

The Java XML DOM wrapper is an abstraction layer over XML parsers that enables you to interpret, create, or modify XML messages before you send them into or out of the integration gateway. The XML DOM wrapper provides an API that is equivalent to the PeopleCode API to support composing and transforming XML documents.

For example, suppose that you need to send incoming XML messages to PeopleSoft Integration Broker over HTTP and that you need to read the content of the messages as they come in. Rather than parse each message on a character-by-character basis or use a parser, you can use the Java XML DOM wrapper to read the XML messages as they come into the integration gateway, add information to them as necessary, and pass the messages on to the integration engine.

You can also use the Java XML DOM wrapper to refine XML messages before they are sent, for example, by changing tag names, verifying information in non-XML format, and so forth.

Use the Java XML DOM wrapper to:

- Navigate through XML documents by using methods, such as GetParentNode, GetChild, and getSibling.
- Control the order of elements within XML documents.
- Support additional XML features, such as namespaces and processing instructions.

Using Java XML DOM Wrapper Classes

The following table lists the Java XML DOM wrapper classes. The objects in these classes provide the basic methods for accessing and modifying DOM objects.

Class	***Description***
XmlDocument	Contains the DOM that supports serialization and deserialization. Provides equivalent methods for dealing with XML documents as PeopleCode does on the application server side.

Class	Description
XmlNode	Provides equivalent methods for dealing with XML nodes as PeopleCode does on the application server side.
XmlNodeList	Provides equivalent methods for dealing with XMLNodeList as PeopleCode does on the application server side.
XmlException	Reports errors that occur while handling the Java XML DOM wrapper classes. These are thrown when an XML DOMException is caught in DOM level. Most of XmlDocument/XmlNode methods throw the XmlException.

Java XML DOM Code Example

The following is a Java XML DOM code example.

```
public String simulateSendingMessage(String message) throws
                        GeneralFrameworkException,
                        DuplicateMessageException,
                        InvalidMessageException,
                        ExternalSystemContactException,
                        ExternalApplicationException,
                        MessageMarshallingException,
                        MessageUnmarshallingException,
                        XmlException {

//Create an XmlDocument object.

//Instantiate a XmlDocument object first. This step is mandatory.
XmlDocument xmlDoc = new XmlDocument();
XmlNode rootNode = null;

//Parse the string into the XmlDocument object.
try {

   //Fill in the XML structure/data with the real XML string
   xmlDoc.ParseXmlFromString(message);

   //Get the root XmlNode
   rootNode = xmlDoc.GetDocumentElement();
   } catch (XmlException xe) {
   throw new InvalidMessageException
   ("ExampleTargetConnector:InvalidMessageException",xe);
}

//Gather credentials from the Xml Document.
XmlNode itemNode;
XmlNode tmpNode;
float price;
float totalPrice = 0;

//Read all Message Items and calculate the totalPrice.
//Get the count of root XmlNode's immediate child XmlNode
for(int i=0; i < rootNode.GetChildNodeCount(); i++) {

//Get the number i child XmlNode
itemNode = rootNode.GetChildNode(i);

// Only process the Item nodes (any other tag will not be //processed).

//Get current XmlNode name
if (!itemNode.GetNodeName().equals("Item")) {
   continue;
}

if (itemNode == null) {
   String[] parms = {"Item"};
   throw new InvalidMessageException("ExampleTargetConnector:
     InvalidMessageException", new MessageCatalogEntry(10503,parms),null);
}

tmpNode = itemNode.FindNode("Price");

   price = Float.parseFloat(tmpNode.GetNodeValue());

   totalPrice += price * quantity;
}

return "<?xml version=\"1.0\"?>"+
   "<ExampleResponse>"+
```

```
   "<Status>"+
      "<Code>0</Code>"+
   "</Status>"+
   "<ResponseBody>"+
      "<TotalPrice>"+totalPrice+"</TotalPrice>"+
   "</ResponseBody>"+
   "</ExampleResponse>";
}
```

See Also

Enterprise PeopleTools 8.50 PeopleBook: PeopleCode API Reference, "XmlDoc Class"

Developing and Implementing Connectors in the C/C++ Environment

You should use Java for connector development. However, the PeopleSoft system also provides an environment for developing connectors using C/C++ through the Java Native Interface (JNI).

This section provides an overview of the development process and discusses how to create target connectors for the C/C++ environment.

Understanding the Development Process

To develop connectors for the C/C++ environment, you typically use Xalan and Xerces by Apache, Inc.

Xalan is an XSLT processor that transforms XML documents into HTML, text, or other XML document types. Xalan uses two jar files: xalan.jar and xerces.jar. Xerces is a Java parser to parse XML.

See http://www.apache.org..

To develop connectors and not use Xalan and Xerces, you can call a PeopleSoft-delivered Java connector, copy what you need from the message, and pass the information to the C/C++ environment.

To develop connectors for C/C++ environments:

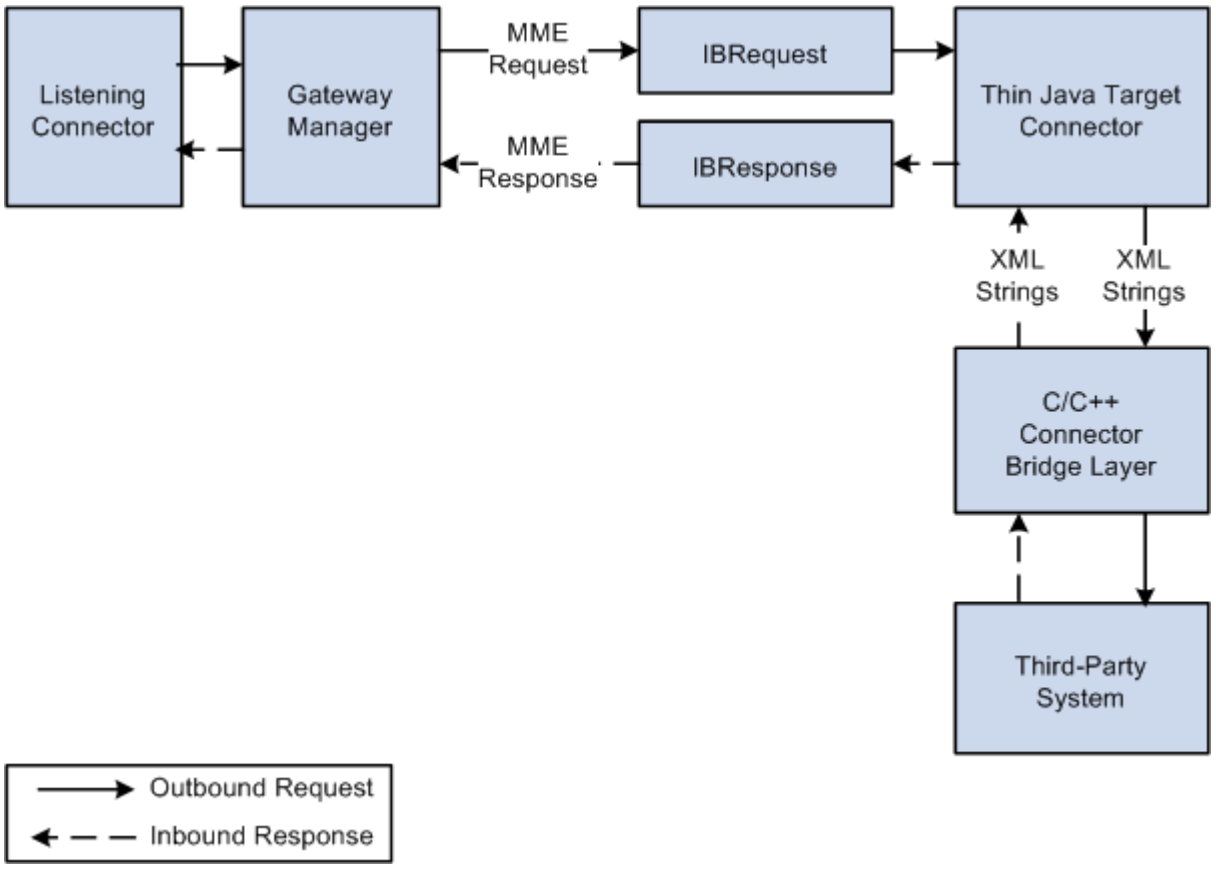
1. Create a Java target connector class.
2. Create a JNI header file.
3. Implement JNI header functions.
4. Build a dynamic-link library (DLL) for the native library.
5. Register the DLL.

Java Target Connectors

Building and implementing a connector in the C/C++ environment includes building a thin Java target connector to enable the gateway manager to load the connector the same way that it loads Java connectors, by parsing the IBRequest object.

The thin Java connector is also a gateway to the C/C++ connector. It is responsible for loading the connector and passing the XML string with business logic to one or more native C/C++ methods.

This diagram illustrates the architecture for implementing target connectors in the C/C++ environment for communication with the PeopleSoft Integration Broker gateway:



Architecture for implementing target connectors in the C/C++ environment

The XML string contains the body from the MIME request string.

The IBResponse object is completely transparent to the native connector. All of the information that the native side needs goes through the XML string or by strings that come from the IBRequest.

PeopleSoft provides a template that you can use as a starting point for developing the connector. In most cases you do not need to make any additions to the code because the IBRequest provides the IBInfo and content body XML strings. However, you can modify the IBInfo and content body XML strings in the C/C++ connector library and declare additional native methods as necessary in the psnativeconnector section of the template.

JNI Headers

After you create a Java connector, you must create a JNI header file by using the javah command. The javah command creates a C/C++ declaration. The JNI header file serves as a bridge between native methods that you call within the Java target connector and those in the C/C++ environment.

JNI Header Functions

When you implement the JNI Header functions, you pass the IBInfo by using a native C/C++ functional call from the Java environment to the third-party system. In doing so, you pass the business logic to the C/C++ system.

DLLs for the Native Library

To build a DLL for the native library, compile the C/C++ functions that are generated by the previous steps.

DLL Registration

Register the DLL that you built for the native library by adding it to the system variables or by adding it to the web server path.

Creating Target Connectors for the C/C++ Environment

To create a target connector for the C/C++ environment:

1. Create a Java target connector.

   a. Copy the code from the following Java target connector template into a text editor:

```
public class CPlusPlusTargetConnector implements TargetConnector {
      // Native method Declaration
      public native String native_simple(String IBReqString,
      String[] xmlStringArray);

      public IBResponse send(IBRequest request)  {

      IBResponse response = null;

      try {
         String ibReqString = request.getIBInfoString();
         String requestXml1 = request.getContentSectionAt(0);
         String requestXml2 = request.getContentSectionAt(1);
         . . .

         // Assign to String[] xmlStringArray
         // Load native lib that implements the connector
         System.loadLibrary("psnativeconnector");

         responseStr = native_simple(ibReqString,
         xmlStringArray);

         response = new IBResponse();
         response.addContentSection(null, responseStr);
   }
   catch (Exception ioe) {
         throw new GeneralFrameworkException(ioe.getMessage());
   }
   return response;
}
```

   b. Compile the code.

      By using a Java compiler, compile the CPlusPlusTargetConnector.java file into a CPlusPlusTargetConnector.class file. Then copy the class file into the target connector directory on the web server. The CLASSPATH environment variable should point to the Integration Broker.jar file.

   c. Install the connector.

   d. Register the connector.

2. Create a JNI header file.

   At a DOS prompt or UNIX shell command prompt, enter the following command and press Enter:

```
hisdir>javah −jni com.peoplesoft.peoplesoft.pt.integrationgateway.
  targetconnector.CplusPlusTargetConnector
```

3. Implement the JNI header file.

   a. Copy the method declaration output from the previous step to a C++ file.

      The following code shows sample method declaration output from the javah command:

      ```
      JNIEXPORT jstring JNICALL Java_com_peoplesoft_pt_
      integrationgateway_targetconnector_CPlusPlusTargetConnector_
      native_1simple (JNIEnv *env, jobject obj, jstring ibInfo,
      jobjectArray contentArr):
      ```

   b. Convert the jstring to an ANSII string or to a Unicode string.

      To convert the jstring to ANSII, follow this example:

      ```
      Const char* string;
      string = env->GetStringCharsUTF(jstrXml, NULL);
      ```

      To convert the jstring to Unicode, follow this example:

      ```
      const TCHAR * tStr;
      tStr = env->GetStringChars(jstrXml, NULL);
      ```

      You can now parse the XML as necessary.

4. Build the PSNativeConnector DLL.

   a. Compile the C++ functions from the previous step into a DLL file.

   b. Save the file.

      You do not have to use PSNativeConnector.DLL as the name for the file, however the name that you use *must*match the connector name in the connector class file. If you use another name for the DLL, enter the new name for the connector in the following line of the connector class file:

      ```
      System.loadLibrary("psnativeconnector");
      ```


5. Register the DLL.

   Register the DLL by adding it to the system variables or by adding it to the path of the web server environment.

   To add the DLL path to the system variables, select Control Panel, System, Environment.

   To add the DLL path to the web server environment, follow the instructions for the web server that you are using:

   - For Oracle WebLogic, open a command prompt or shell command and append the path to the library at the end of the following line:

     ```
     :runWebLogic
     echo on
     set PATH=.\bin;%PATH%;%PATH_TO_YOUR_NATIVE_LIBRARY%
     ```

   - For IBM WebSphere, open a command prompt or shell command, type the following, and press Enter:

     ```
     C:\Apps\WebSphere\AppServer\bin\startServer.bat
     ```

     Then, append the path to the library in this line:

     ```
     SET PATH=;%PATH_TO_YOUR_NATIVE_LIBRARY%;%WAS_HOME%/
     bin;%JAVA_HOME%/bin;%JAVA_HOME%/jre/bin;%PATH%
     ```

See Also

Chapter 5, "Managing Integration Gateways," Loading Target Connectors, page 35

Appendix B, "Using the Integration Broker Connector SDK," Installing Connector Classes, page 315

Appendix B, "Using the Integration Broker Connector SDK," Registering Connectors, page 315

Reusing Connector Code

This section discusses how to:

- Reuse connector code through inheritance.
- Reuse connector code through delegation.

Reusing Connector Code Through Inheritance

Use inheritance to extend an existing connector and provide additional behavior.


```
public Class MyHttpTargetConnector   extends HttpTargetConnector {
public IBResponse send(IBRequest request) throws
                     GeneralFrameworkException,
                     InvalidMessageException,
                     ExternalSystemContactException,
                     ExternalApplicationException,
                     MessageMarshallingException,
                     MessageUnmarshallingException,
                     DuplicateMessageException  {

   // Do connector specific logic here possibly modifying the
   // request . . .
   IBResponse response = super.send(request);

   // Do connector specific logic here possibly modifying the
   // response . . .
   }
}
```

Reusing Connector Code Through Delegation

By using delegation, you can reuse the code from an existing connector to create a new connector, as shown in this example:


```
public Class MyHttpTargetConnector   implements TargetConnector {
   public IBResponse ping(IBRequest request) throws
                      GeneralFrameworkException,
                      InvalidMessageException,
                      ExternalSystemContactException,
                      ExternalApplicationException,
                      MessageMarshallingException,
                      MessageUnmarshallingException,
                      DuplicateMessageException  {

   HttpTargetConnector httpTargetConnector = new
      HttpTargetConnector();
   return httpTargetConnector.ping(request);
}
public IBResponse send(IBRequest request) throws
                      GeneralFrameworkException,
                      InvalidMessageException,
                      ExternalSystemContactException,
                      ExternalApplicationException,
                      MessageMarshallingException,
                      MessageUnmarshallingException,
                      DuplicateMessageException  {

   // Do connector specific logic here possibly modifying the
   // request . . .
   HttpTargetConnector httpTargetConnector = new
      HttpTargetConnector();
   IBResponse response = httpTargetConnector.send(request);

   // Do connector specific logic here possibly modifying the
   // response . . .
}

public ConnectorDataCollection introspectConnector() {
   HttpTargetConnector httpTargetConnector = new
      HttpTargetConnector();
   ConnectorDataCollection cdc =
      httpTargetConnector.introspectConnector();

   // Possibly add specific fields here . . .
   Return cdc;
   }
}
```

Index

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X