ORACLE
PEOPLESOFT ENTERPRISE

Enterprise PeopleTools 8.49 PeopleBook: PeopleSoft Integration Broker

March 2007

ORACLE

Enterprise PeopleTools 8.49 PeopleBook: PeopleSoft Integration Broker
SKU PT849IBR-B 0307
Copyright © 1988-2007, Oracle. All rights reserved.

The Programs (which include both the software and documentation) contain proprietary information; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent, and other intellectual and industrial property laws. Reverse engineering, disassembly, or decompilation of the Programs, except to the extent required to obtain interoperability with other independently created software or as specified by law, is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. This document is not warranted to be error-free. Except as may be expressly permitted in your license agreement for these Programs, no part of these Programs may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose.

If the Programs are delivered to the United States Government or anyone licensing or using the Programs on behalf of the United States Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS

Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the Programs, including documentation and technical data, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement, and, to the extent applicable, the additional rights set forth in FAR 52.227-19, Commercial Computer Software--Restricted Rights (June 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065.

The Programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be the licensee's responsibility to take all appropriate fail-safe, backup, redundancy and other measures to ensure the safe use of such applications if the Programs are used for such purposes, and we disclaim liability for any damages caused by such use of the Programs.

The Programs may provide links to Web sites and access to content, products, and services from third parties. Oracle is not responsible for the availability of, or any content provided on, third-party Web sites. You bear all risks associated with the use of such content. If you choose to purchase any products or services from a third party, the relationship is directly between you and the third party. Oracle is not responsible for: (a) the quality of third-party products or services; or (b) fulfilling any of the terms of the agreement with the third party, including delivery of products or services and warranty obligations related to purchased products or services. Oracle is not responsible for any loss or damage of any sort that you may incur from dealing with any third party.

Oracle, JD Edwards, PeopleSoft, and Siebel are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

Open Source Disclosure

Oracle takes no responsibility for its use or distribution of any open source or shareware software or documentation and disclaims any and all liability or damages resulting from use of said software or documentation. The following open source software may be used in Oracle's PeopleSoft products and the following disclaimers are provided.

Apache Software Foundation

This product includes software developed by the Apache Software Foundation (http://www.apache.org/). Copyright © 2000-2003. The Apache Software Foundation. All rights reserved. Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at http://www.apache.org/licenses/LICENSE-2.0.

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

OpenSSL

Copyright © 1998-2005 The OpenSSL Project. All rights reserved.

This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (http://www.openssl.org/).

THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE OpenSSL PROJECT OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Loki Library

Copyright © 2001 by Andrei Alexandrescu. This code accompanies the book: Alexandrescu, Andrei. "Modern C++ Design: Generic Programming and Design Patterns Applied". Copyright © 2001 Addison-Wesley. Permission to use, copy, modify, distribute and sell this software for any purpose is hereby granted without fee, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation.

Helma Project

Copyright © 1999-2004 Helma Project. All rights reserved. THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE HELMA PROJECT OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Helma includes third party software released under different specific license terms. See the licenses directory in the Helma distribution for a list of these license.

Sarissa

Copyright © 2004 Manos Batsis.

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA.

ICU

ICU License - ICU 1.8.1 and later COPYRIGHT AND PERMISSION NOTICE Copyright © 1995-2003 International Business Machines Corporation and others. All rights reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, provided that the above copyright notice(s) and this permission notice appear in all copies of the Software and that both the above copyright notice(s) and this permission notice appear in supporting documentation. THE SOFTWARE IS PROVIDED "AS IS," WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE BE LIABLE FOR ANY CLAIM, OR ANY SPECIAL INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE. Except as contained in this notice, the name of a copyright holder shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization of the copyright holder.

All trademarks and registered trademarks mentioned herein are the property of their respective owners.

Sun's JAXB Implementation – JDSDK 1.5 relaxngDatatype.jar 1.0 License

Copyright © 2001, Thai Open Source Software Center Ltd, Sun Microsystems. All rights reserved.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

W3C IPR SOFTWARE NOTICE

Copyright © 2000 World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University). All Rights Reserved.

Note: The original version of the W3C Software Copyright Notice and License could be found at http://www.w3.org/Consortium/Legal/copyright-software-19980720.

THIS SOFTWARE AND DOCUMENTATION IS PROVIDED "AS IS," AND COPYRIGHT HOLDERS MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF THE SOFTWARE OR DOCUMENTATION WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS. COPYRIGHT HOLDERS WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF ANY USE OF THE SOFTWARE OR DOCUMENTATION.

Contents

General Preface

About This PeopleBook xxxv

PeopleSoft Enterprise Application Prerequisites xxxv

Application Fundamentals xxxv

Documentation Updates and Printed Documentation xxxvi

Obtaining Documentation Updates xxxvi

Downloading and Ordering Printed Documentation xxxvi

Additional Resources xxxvii

Typographical Conventions and Visual Cues xxxviii

Typographical Conventions xxxviii

Visual Cues xxxix

Country, Region, and Industry Identifiers xxxix

Currency Codes xl

Comments and Suggestions xl

Common Elements Used in PeopleBooks xl

Preface

PeopleSoft Integration Broker Preface xliii

PeopleSoft Integration Broker xliii

Chapter 1

Getting Started with PeopleSoft Integration Broker 1

PeopleSoft Integration Broker Overview 1

Implementing PeopleSoft Integration Broker 1

Other Sources of Information 4

Chapter 2

Understanding PeopleSoft Integration Broker 5

Introduction to PeopleSoft Integration Broker 5

Web Services 5

Integration Gateway 6

Integration Engine 6

Copyright © 1988-2007, Oracle. All rights reserved.

Integration Gateway Architecture.....7
Architecture Elements.....7
Connectors.....8
Gateway Manager.....9
Gateway Services.....9
Integration Engine Architecture.....10
Service Operations and Messages.....11
Service Operation Types.....12
Operation Types.....12
Incoming and Outgoing Request Flows.....14
Incoming Request Flow.....14
Outgoing Request Flow.....17

Chapter 3
Understanding Messaging.....19
Asynchronous Messaging.....19
Brokers, Contractors and Queues.....19
Messaging System Server Processes.....20
Dispatchers and Handlers.....21
Asynchronous Service Operation Publication.....22
Asynchronous Service Operation Subscription.....25
Synchronous Messaging.....28
Synchronous Service Operation Publication.....28
Synchronous Service Operation Subscription.....29

Chapter 4
Understanding Creating and Implementing Integrations.....31
Determining the Messaging Architecture.....31
Installing Web Servers.....32
Installing PeopleTools.....32
Installing Application Databases.....32
Starting the PeopleSoft Pure Internet Architecture.....32
Configuring and Starting Messaging Servers for Asynchronous Messaging.....33
Activating Pub/Sub Server Domains.....33
Defining Integration Gateways and Loading Connectors.....33
Configuring Integration Gateway Properties.....34
Configuring PeopleSoft Integration Broker to Handle Services.....34
Creating Integration Metadata.....35

Copyright © 1988-2007, Oracle. All rights reserved.

Understanding Integration Metadata.....35
Order of Precedence for Creating Integration Metadata.....36
Granting Security Access Service Operations.....36

Chapter 5
Using the Integration Broker Quick Configuration Page.....37
Prerequisites for Using the Integration Broker Quick Configuration Page.....37
Accessing the Integration Broker Quick Configuration Page.....37

Chapter 6
Administering Messaging Servers for Asynchronous Messaging.....41
Understanding Messaging Server Administration.....41
Messaging Servers.....41
Messaging Servers in the DB2 UDB OS/390 and z/OS Environments.....42
Messaging Server Processes.....42
Understanding Dedicated Messaging Servers.....43
Considerations When Creating Dedicated Servers.....45
Creating and Assigning Dedicated Servers.....45
Editing Messaging Server Queue Lists.....47
Deleting Messaging Servers.....48
Configuring Messaging Servers.....48
Specifying Dispatcher Parameters.....48
Specifying Handler Parameters.....51
Setting the BEA Tuxedo Queue Size.....52

Chapter 7
Managing Integration Gateways.....53
Understanding Integration Gateway Configuration.....53
Local Gateway Compatibility.....53
Types of Integration Gateway Configuration.....53
The Gateways Component.....54
Minimum Integration Gateway Setup Requirements.....54
Administering Integration Gateways.....54
Pages Used to Administer Integration Gateways.....55
Defining Integration Gateways.....55
Pinging Integration Gateways.....57
Loading Target Connectors.....57

Copyright © 1988-2007, Oracle. All rights reserved.

Refreshing Integration Gateway Properties....58
Editing Connector Properties....58
Accessing Gateway Setup Properties....60
Page Used to Access Integration Gateway Properties....60
Accessing Gateway Properties....60
Setting BEA Jolt Connection Properties....61
Understanding BEA Jolt Connection Properties....61
Page Used to Set BEA Jolt Connection Properties....62
Setting BEA Jolt Connection String Properties....62
Using the integrationGateway.properties File....64
Accessing the integrationGateway.properties File....64
Entering Values in the integrationGateway.properties File....65
Encrypting Passwords....66
Encrypting Passwords in the PeopleSoft Pure Internet Architecture....66
Encrypting Passwords Using the PSCipher Java Utility....66
Configuring Security and General Properties....67
Understanding Integration Gateway Properties and OAS Clustering....67
Setting Security Properties....67
Specifying the Gateway Version....68
Specifying the Gateway Class Location....69
Setting General Connection Properties....69
Setting Logging Properties....72
Setting DTD Validation Properties....73
Setting BEA Jolt Session Pooling Parameters....74
Applying Message Transformations at the Integration Gateway....74
Understanding Applying Message Transformations at the Integration Gateway....74
Developing and Implementing Gateway-Based Transformation Programs....75
Setting Integration Gateway Properties for Gateway-Based Transformations....75
Understanding Logged Errors....77
Bypassing Integration Engines to Send Messages....78
Using the ConnectorRequest Built-In Function....78
Using the ConnectorRequestURL Built-In Function....78

Chapter 8
Understanding Supported Message Structures....81
Integration Broker Message Structures....81
Internal Message Format for Request Messages....81
Internal Message Format for Response Messages....91
Local Compression....94

Copyright © 1988-2007, Oracle. All rights reserved.

Accessing IBInfo Elements Using PeopleCode.....95
PeopleSoft Rowset-Based Message Format.....96
Understanding the PeopleSoft Rowset-Based Message Format.....96
Rowset-Based Message Template.....97
FieldTypes Section.....97
MsgData Section.....98
PSCAMA.....99
Identifying Changes to Field-Level Attributes.....102
PeopleSoft Timestamp Format.....102
Schema Restrictions.....102
Rowset-Based Message Example.....103
Nonrowset-Based Message Structures.....105
XML DOC-Compliant Messages.....105
SOAP-Compliant Messages.....106
Non-XML Files.....106
Using Nonrowset-Based Messages in Service Operations Exposed as WSDL.....108
Message Parts Structures.....108
Understanding Message Part Structures.....108
Rowset-Based Message Parts.....108
Nonrowset-Based Message Parts.....111
Message Container Structures.....111
Example 1: XML Schema of a Container Message with Rowset-Based Message Parts.....112
Example 2: XML Schema of a Container Message with Nonrowset-Based Message Parts.....112

Chapter 9
Using Listening Connectors and Target Connectors.....115
Understanding Listening Connectors and Target Connectors.....115
Listening Connectors.....115
Target Connectors.....117
Working With the PeopleSoft Connectors.....122
Understanding the PeopleSoft Connectors.....122
Using the PeopleSoft Listening Connector.....122
Using the PeopleSoft Target Connector.....122
Working With the HTTP Connectors.....123
Understanding the HTTP Connectors.....123
Using the HTTP Listening Connector.....123
Using the HTTP Target Connector.....126
Complying With Message Formatting and Transmission Requirements.....129
Understanding HTTP Status Codes.....136

Copyright © 1988-2007, Oracle. All rights reserved.

Running Integration Gateways Behind Proxy Servers....137
Working With the PeopleSoft Services Listening Connector....137
Understanding the PeopleSoft Services Listening Connector....138
Setting Parameters for the PeopleSoft Services Listening Connector....138
Passing Parameters to the PeopleSoft Services Listening Connector....138
Passing Parameters to Get XML Schema, WSDL and WSIL....139
Working With the PeopleSoft 8.1 Connectors....139
Understanding the PeopleSoft 8.1 Connectors....139
Using the PeopleSoft 8.1 Listening Connector....140
Using the PeopleSoft 8.1 Target Connector....140
Working With the JMS Connectors....141
Understanding the JMS Connectors....141
Specifying JNDIFactory Class Names....142
Using the JMS Listening Connector....142
Using the JMS Target Connector....149
Adding Generic JMS Providers....155
Working With the Simple File Target Connector....156
Understanding the Simple File Target Connector....156
Setting File Security....156
Node-Level Connector Properties....156
Working With the FTP Target Connector....157
Understanding the FTP Target Connector....157
Prerequisites for Using the FTP Target Connector....158
Specifying Required JAR Files....158
Setting Node-Level FTP Connector Properties....158
Setting Node-Level FTPS Connector Properties....160
Using Directory Lists....161
Directory List Example....162
Working With the AS2 Connectors....164
Understanding Using AS2....164
Understanding MDNs....164
PeopleCode Considerations....165
Understanding the AS2 Listening Connector....165
Understanding the AS2 Response Connector....166
Understanding the AS2 Target Connector....166
Using the AS2 Listening Connector....167
Using the AS2 Target Connector....169
Working With the SMTP Target Connector....174

Copyright © 1988-2007, Oracle. All rights reserved.

Chapter 10
Managing Messages.....177
Understanding Managing Messages.....177
Message Definitions.....177
Message Types.....177
Message Record Structure.....178
Underlying Record Definitions.....178
Fields Defined as Uppercase.....178
Restrictions for Modifying Messages.....178
Adding Message Definitions.....179
Understanding Adding Message Definitions.....179
Page Used to Add Message Definitions.....179
Adding a Message Definition.....179
Managing Rowset-Based Messages.....181
Understanding Managing Rowset-Based Messages.....181
Pages Used to Manage Rowset-Based Messages.....182
Inserting Root Records.....182
Inserting Child and Peer Records.....183
Specifying Record Aliases.....184
Deleting Records.....184
Excluding Fields from Messages.....184
Specifying Field Name Aliases.....185
Generating XML Message Schemas for Rowset-Based Messages.....185
Managing Nonrowset-Based Messages.....186
Understanding Managing Nonrowset-Based Messages.....187
Page Used to Manage Nonrowset-Based Messages.....187
Adding XML Message Schemas to Nonrowset-Based Messages.....187
Editing Nonrowset-Based XML Schemas.....187
Managing Message Parts.....188
Understanding Message Parts.....188
Creating Part Messages.....188
Managing Container Messages.....189
Understanding Managing Container Messages.....189
Pages Used to Manage Container Messages.....189
Adding Message Parts to Container Messages.....189
Generating XML Message Schemas for Container Messages.....193
Renaming and Deleting Message Definitions.....194
Pages Used to Rename and Delete Message Definitions.....195
Renaming Message Definitions.....195
Deleting Messages Definitions.....196

Copyright © 1988-2007, Oracle. All rights reserved.

Deleting Messages During Upgrade....196

Chapter 11
Managing Service Operation Queues....197
Understanding Service Operation Queues....197
Adding Queue Definitions....197
Page Used to Create Queue Definitions....197
Adding a Queue Definition....197
Applying Queue Partitioning....199
Understanding Queue Partitioning....199
Selecting Partitioning Fields....200
Renaming and Deleting Queues....201
Pages Used to Rename and Delete Queue Definitions....202
Renaming Queue Definitions....202
Deleting Queue Definitions....203
Deleting Queues During Upgrade....203

Chapter 12
Sending and Receiving Messages....205
Understanding Sending and Receiving Messages....205
Prerequisites for Sending and Receiving Messages....205
Messaging Process Flows....205
Understanding Integration PeopleCode....207
Sending and Receiving PeopleCode....207
Application Classes....208
Routing Methods....209
Messaging Methods....212
Messaging PeopleCode....218
Messaging Handlers....218
Selecting Handlers....219
Implementing Handlers....220
Generating and Sending Messages....221
Understanding Outbound Messaging....222
Handling Outbound Asynchronous Message Transmission....223
Handling Outbound Synchronous Transactions....225
Overriding Synchronous Timeout Intervals at Runtime....227
Handling Cookies....227
Setting and Overriding Target Connector Properties at Runtime....228

Copyright © 1988-2007, Oracle. All rights reserved.

Receiving and Processing Messages.....232
Handling Inbound Asynchronous Transactions.....232
Handling Inbound Synchronous Transactions.....247
Simulating Receiving Messages from External Nodes.....250
Processing Inbound Errors.....251
Validating Data.....251
Using the Exit Built-in Function.....252
Using Message Object Functionality With Nonrowset-Based Messages.....253
Using the SetXMLDoc Method.....254
Using the GetXMLDoc Method.....254
Generating Test Messages.....254
Working With Message Segments.....254
Understanding Message Segments.....255
Understanding PeopleCode used to Work with Message Segments.....255
Configuring Nodes to Handle Segmented Messages.....256
Creating Message Segments.....257
Deleting Message Segments.....259
Sending and Receiving Segmented Messages.....260
Accessing Segments in Messages.....260
Viewing Message Segment Data.....262
Using Restartable Processing for Publishing Large Messages in Batch.....262

Chapter 13
Building Message Schemas.....265
Understanding the Message Schema Builder.....265
Message Schemas.....265
Building, Importing, Modifying and Deleting Message Schemas.....265
Selecting and Viewing Data in the Message Schema Builder.....266
Pages Used To Select and View Data in the Message Schema Builder.....266
Selecting Data in the Message Schema Builder.....267
Viewing Message Schema Details.....268
Viewing XML Message Schema.....269
Building Message Schemas for Rowset-Based Messages.....269
Page Used to Build Message Schemas for Rowset-Based Messages.....270
Building a Message Schema for a Rowset-Based Message.....270
Importing Message Schemas for Nonrowset-Based Messages.....270
Pages Used to Import Message Schemas for Nonrowset-Based Messages.....270
Importing a Message Schema for a Nonrowset-Based Message.....270
Modifying Message Schemas.....271

Copyright © 1988-2007, Oracle. All rights reserved.

Pages Used to Modify Message Schemas....271
Modifying a Message Schema....271
Deleting Message Schemas....272
Understanding Deleting Message Schemas....272
Page Used to Delete Message Schemas....273
Using the Message Schema Builder Page to Delete Message Schemas....273
Using the Message Schemas Page to Delete Message Schemas....273

Chapter 14
Managing Services....275
Understanding Managing Services....275
Common Elements Used in This Chapter....275
Configuring PeopleSoft Integration Broker for Handling Services....277
Understanding Configuring PeopleSoft Integration Broker for Handling Services....277
Page Used to Configuring PeopleSoft Integration Broker for Handling Services....279
Setting Service Configuration Properties....279
Specifying UDDI Repositories in the PeopleSoft System....280
Understanding Specifying UDDI Repositories in the PeopleSoft System....280
Page Used to Specify UDDI Repositories in the PeopleSoft System....281
Specifying UDDI Repositories in the PeopleSoft System....281
Accessing and Viewing Service Definitions....282
Pages Used to Access and View Service Definitions....282
Accessing Service Definitions....282
Viewing WSDL Documents Generated for Services....283
Viewing Service Operation Information....284
Viewing Messages Defined for Service Operations....284
Adding Service Definitions....284
Page Used to Add Service Definitions....284
Adding Services....285
Configuring Service Definitions....285
Page Used to Configure Service Definitions....285
Configuring a Service Definition....285
Restricting and Enabling Write Access to Services....287
Understanding Restricting Write Access to Services....287
Page Used to Restrict and Enable Write Access to Services....288
Restricting Write Access to Services....288
Enabling Write Access to Services....288
Renaming and Deleting Services....289
Page Used to Rename and Delete Services....290

Copyright © 1988-2007, Oracle. All rights reserved.

Renaming Services....290
Deleting Services....290

Chapter 15
Managing Service Operations....291
Understanding Managing Service Operations....291
Service Operations....291
Service Operation Types....291
Service Operation Aliases....292
Service Operation Versions....292
Mapping Service Operations....292
Accessing and Viewing Service Operation Definitions....292
Pages Used to Access and View Service Operation Definitions....293
Accessing Service Operation Definitions....293
Viewing Service Operation Definitions....295
Adding Service Operation Definitions....298
Page Used to Add Service Operation Definitions....298
Adding a Service Operation Definition....298
Configuring Service Operation Definitions....298
Pages Used to Configure Service Operation Definitions....299
Specifying General Information....299
Defining Service Operation Version Information....300
Adding Handlers to Service Operations....301
Adding Routing Definitions....303
Activating and Inactivating Routing Definitions....304
Setting Permissions to Service Operations....304
Understanding Setting Permission to Service Operations....304
Page Used to Set Permissions to Service Operations....304
Setting Permission Access to Service Operations....305
Changing the Services with Which Service Operations are Associated....305
Page Used to Change the Services with Which Service Operations are Associated....305
Changing the Service with Which a Service Operation is Associated....305
Managing Service Operation Versions....306
Page Used to Manager Service Operation Versions....307
Creating Service Operation Versions....307
Using Non-Default Service Operation Versions....307
Attaching Files to Service Operations....308
Understanding Attaching Files to Service Operations....308
Page Used to Attach Files to Service Operations....308

Copyright © 1988-2007, Oracle. All rights reserved.

Using the FTP Attachment Utility....308
Sending Attachment Information with Service Operations....309
Processing Attachment Information Included in Service Operations....310
Renaming and Deleting Service Operations....311
Page Used to Rename and Delete Service Operations....312
Renaming Service Operations....312
Deleting Service Operations....312

Chapter 16
Enabling Runtime Message Schema Validation....315
Understanding Message Schema Validation....315
Message Schema Validation....315
Prerequisites for Validating Message Schemas....315
Selecting Service Operations....316
Pages Used to Select Service Operations....316
Selecting a Service Operation....316
Viewing Defined Message Schemas....318
Pages Used to View Defined Message Schemas....318
Viewing XML Schemas Defined for Messages....318
Enabling Runtime Message Schema Validation....319
Page Used to Enable Runtime Message Schema Validation....319
Enabling Runtime Message Schema Validation....320

Chapter 17
Creating Component Interface-Based Services....321
Understanding Creating Component Interface-Based Services....321
Naming Conventions Integration Metadata Created....321
User-Defined Method Restrictions....322
Impact of Changing Component Interfaces....323
Prerequisites....323
Selecting Component Interfaces to Expose as Services....323
Page Used to Select Component Interfaces....323
Selecting Component Interfaces....323
Selecting Component Interface Methods to Include as Service Operations....324
Page Used to Select Methods to Include as Service Operations....325
Selecting Methods to Include in Service Operations....325
Generating Component Interface-Based Services....327
Page Used to Generate Component Interfaced-Based Services....327

Copyright © 1988-2007, Oracle. All rights reserved.

Generating Services and Service Operations....327
Viewing Component Interface-Based Service Definitions....328
Pages Used to View Component Interface-Based Service Definitions....328
Viewing Component Interface-Based Service Definitions....328

Chapter 18
Managing Routing Definitions....331
Understanding Routing Definitions....331
Routing Definitions....331
Routing Types....331
Defining Routing Definitions....332
Methods for Generating and Defining Routing Definitions....332
Routing Definition Naming Conventions....333
Routing Definition External Aliases....334
Service Operation Mapping....334
Viewing Routing Definitions....335
Managing System-Generated Routing Definitions....335
Understanding Managing System-Generated Routing Definitions....335
Page Used to Manage System-Generated Routing Definitions....335
Viewing System-Generated Routing Definition Status....336
Initiating System-Generated Routing Definitions....336
Regenerating System-Generated Routing Definitions....337
Creating Routing Definitions....338
Understanding Creating Routing Definitions....338
Pages Used to Create Routing Definitions....340
Adding Routing Definitions....340
Defining General Routing Information....342
Viewing Routing Parameters for Requests and Responses....344
Overriding Gateway and Connector Properties....346
Using Introspection to Create Routing Definitions....347
Understanding Using Introspection to Create Routing Definitions....347
Prerequisites for Using Introspection to Create Routing Definitions....348
Pages Used to Using Introspection to Create Routing Definitions....348
Selecting Service Operations for Which to Create Routing Definitions....348
Selecting Nodes to Introspect....350
Selecting Routing Definitions to Generate....351
View Introspection Results....353
Activating and Inactivating Routing Definitions....354
Understanding Activating and Inactivating Routing Definitions....354

Copyright © 1988-2007, Oracle. All rights reserved.

Pages Used to Activate and Inactivate Routing Definitions.....355
Activating and Inactivating Routing Definitions in the Routing Component.....355
Activating and Inactivating Routing Definitions in the Service Operations Component.....355
Activating and Inactivating Routing Definitions in the Nodes Component.....355
Renaming and Deleting Routing Definitions.....356
Pages Used to Rename and Delete Routing Definitions.....357
Renaming Routing Definitions.....357
Deleting Routing Definitions.....358
Deleting Duplicate Routing Definitions.....358
Page Used to Delete Duplicate Routing Definitions.....359
Deleting Duplicate Routings.....359

Chapter 19
Adding and Configuring Nodes.....361
Understanding Adding and Configuring Nodes.....361
Prerequisites.....361
Local and Remote Nodes.....361
Adding Node Definitions.....362
Page Used to Add Node Definitions.....362
Adding a Node Definition.....362
Configuring Nodes.....363
Pages Used to Configure Nodes.....363
Defining Node Parameters.....363
Specifying Contact Information.....367
Defining Node Properties.....367
Specifying Gateways and Connectors.....368
Renaming or Deleting Nodes.....370
Understanding Renaming and Deleting Nodes.....370
Page Used to Rename and Delete Nodes.....371
Renaming or Deleting a Node.....371

Chapter 20
Applying Filtering, Transformation and Translation.....373
Understanding Filtering, Transformation, and Translation.....373
Understanding Transform Programs.....373
Transform Programs.....374
Transformation Programming Languages.....374
Third-Party Considerations.....375

Copyright © 1988-2007, Oracle. All rights reserved.

Defining Transform Programs.....375
Understanding Defining Transform Programs.....376
Defining a Transform Program.....376
Developing Transform Programs.....378
Understanding Developing Transform Programs.....378
Inserting Steps and Actions into Transform Programs.....379
Invoking Transform Programs.....380
Renaming or Deleting Transform Programs.....380
Tracing Transform Programs.....380
Accessing Message Data.....380
Making Working Storage Data Available Globally.....382
Preserving Record and Field Aliases.....383
Developing Transformations Using Oracle XSL Mapper.....384
Understanding Oracle XSL Mapper.....384
Development Considerations.....384
Prerequisites.....384
Installing Oracle XSL Mapper.....385
Specifying the Path to the Oracle XSL Mapper Installation Location.....385
Launching Oracle XSL Mapper.....385
Accessing Oracle JDeveloper 10g Documentation and Online Resources.....387
Navigating in Oracle XSL Mapper.....388
Mapping Records and Fields.....390
Deleting Record and Field Maps.....391
Viewing Raw XSLT Code.....392
Testing XSL Maps.....392
Adding and Modifying XSL Map Code.....393
Filtering Messages.....394
Understanding Message Filtering.....395
PeopleCode Filtering Example.....395
Applying Transformations.....397
Understanding Transformation.....397
Using XSLT for Transformation.....397
Performing Data Translation.....399
Understanding Data Translation.....399
Defining Codeset Groups.....400
Defining Codesets.....402
Defining Codeset Values.....402
Importing and Exporting Codesets Between Databases.....404
Deleting Codesets.....404
Using XSLT for Data Translation.....405

Copyright © 1988-2007, Oracle. All rights reserved.

XSLT Translation Example.....407
PeopleCode Translation Example.....410
Terminating Transformation Programs.....412

Chapter 21
Using the Service Operations Monitor.....413
Understanding the Service Operations Monitor.....413
Service Operations Monitor Security.....414
Service Operations Monitor Features and Components.....414
Filtering Asynchronous and Synchronous Service Operations Data.....415
Understanding Filtering Asynchronous and Synchronous Service Operations Data.....415
Selecting Filtering Criteria.....415
Saving Filtering Selections.....416
Monitoring Asynchronous Service Operations.....416
Understanding Monitoring Asynchronous Service Operations Data.....416
Asynchronous Service Operation Statuses.....417
Service Operation Status and Blocked and Stalled Queues.....418
Pages Used to Monitor Asynchronous Service Operations.....419
Filtering Asynchronous Service Operations Data.....419
Viewing Monitor Output for Asynchronous Service Operations Data.....421
Monitoring Service Operation Transactions.....422
Monitoring Asynchronous Service Operation Instances.....423
Monitoring Publication Contracts.....424
Monitoring Subscription Contracts.....425
Viewing Queue Partitioning Information.....426
Viewing Asynchronous Service Operation Details.....427
Common Elements Used to View Asynchronous Service Operation Details.....427
Pages Used to View Asynchronous Service Operation Details.....430
Viewing Asynchronous Service Operation Instance Details.....431
Viewing Asynchronous Publication Contracts Details.....432
Viewing Asynchronous Subscription Contracts Details.....433
Setting the Data Length View Limit for Displaying XML.....434
Monitoring Synchronous Service Operations.....435
Understanding Synchronous Service Operation Statuses.....435
Page Used to View Synchronous Service Operations.....435
Filtering Synchronous Service Operations Data.....436
Viewing Monitor Output for Synchronous Service Operations Data.....437
Viewing Synchronous Service Operation Instance Details.....438
Pages Used to View Synchronous Service Operations Instance Details.....438

Copyright © 1988-2007, Oracle. All rights reserved.

Viewing Synchronous Service Operation Details.....438
Resubmitting and Canceling Service Operations for Processing.....440
Understanding Resubmitting and Canceling Service Operations for Processing.....441
Pages Used to Resubmit and Cancel Service Operations for Processing.....441
Resubmitting and Canceling Individual Service Operations.....441
Resubmitting and Canceling Service Operations in Bulk.....442
Viewing Service Operation IB Info Data.....442
Pages Used to View IB Info Data.....442
Viewing IB Info Data.....442
Viewing Service Operation Errors.....443
Common Elements Used in This Section.....443
Pages Used to View Service Operation Errors.....444
Viewing Asynchronous Service Operation Instance Errors.....444
Viewing Publication Contract Errors.....445
Viewing Asynchronous Subscription Contract Errors.....445
Viewing Synchronous Service Operations Errors.....445
Viewing and Editing Service Operation XML.....446
Understanding Viewing and Editing Service Operation XML.....446
Pages Used to View and Edit Service Operation XML.....447
Viewing Service Operation XML.....447
Editing Service Operation XML.....448
Viewing Service Operation Nonrepudiation Signature Information.....449
Understanding Viewing Service Operation Nonrepudiation Signature Information.....449
Pages Used to View Service Operation Nonrepudiation Signature Information.....449
Viewing Nonrepudiation Signatures in XML Format.....449
Running Batch Error Notification Processes.....450
Understanding Batch Error Notification.....450
Prerequisites for Using Batch Error Notification.....451
Creating Static Error Notification Lists.....451
Running Batch Error Notification.....452
Archiving Service Operation Instances.....453
Prerequisites.....453
Pages Used to Archive Service Operation Instances.....454
Archiving Service Operations.....454
Retrieving Archived Messages.....454
Running Batch Service Operation Archiving Processes.....455
Understanding Running Batch Service Operation Archiving Processes.....455
Prerequisites for Running Batch Service Operation Archiving Processes.....455
Page Used to Run Batch Service Operation Archiving Processes.....455
Running Batch Service Operation Archiving Processes.....455

Copyright © 1988-2007, Oracle. All rights reserved.

Viewing System Performance Statistics....457
Understanding Messaging System Performance Statistics....457
Common Elements Used in this Section....458
Pages Used to View System Performance Statistics....459
Viewing Messaging System Performance Statistics....460
Enabling the Performance Statistics Feature....463
Selecting Statistics Data to View....463
Viewing Inbound Asynchronous Post Statistics....465
Viewing Broker Handler Statistics....466
Viewing Subscription Contract Handler Statistics....467
Viewing Publication Contract Handler Statistics....467
Viewing Inbound Synchronous Service Operation Statistics....470
Viewing Outbound Synchronous Message Statistics....472
Viewing Local Synchronous Service Operation Statistics....474
Managing Pub/Sub Server Domains....477
Understanding Managing Pub/Sub Domains....477
Page Used to Manage Domain Status....477
Working with the Domain Status Page....477
Viewing Dispatcher Status....479
Activating Pub/Sub Server Domains....479
Inactivating Pub/Sub Server Domains....479
Changing Dispatcher Status for Processes....480
Setting Domain Grace Periods....480
Setting Up Domain Failover....480
Understanding Domain Failover....480
Understanding Dynamic and Static Master-Slave Dispatchers....481
Page Used to Set Up Domain Failover....482
Enable Failover on Domains....482
Setting Up Dynamic Master-Slave Dispatchers....483
Checking Queue Set Validity....484
Viewing Queues Assigned to Failover Groups....484
Managing Down Nodes....484
Understanding Managing Down Nodes....484
Page Used to Manage Down Nodes....485
Viewing Transaction Information for Down Nodes....485
Clearing Transaction Data for System Node Restart....485
Pausing, Testing, and Pinging Nodes....486
Understanding Pausing Nodes....486
Page Used to Pause, Test and Ping Nodes....486
Adding Pause Times to Local Nodes....487

Copyright © 1988-2007, Oracle. All rights reserved.

Deleting Pause Times.....487
Testing Local Nodes.....487
Pinging Remote Nodes.....487
Pausing and Starting Queues.....488
Page Used to Pause and Start Queues.....488
Pausing Queues.....488
Starting Queues.....489
Cleaning Up Orphaned Data From Segment Batch Processing Errors.....490
Understanding Cleaning Up Orphaned Data from Segment Batch Process Errors.....490
Page Used to Clean Up Orphaned Data from Segment Batch Processing.....490
Cleaning Up Orphaned Data from Segment Batch Processing Jobs.....490
Using Custom-Defined Components to View Service Operations Data.....491
Understanding Using Custom-Defined Components to View Service Operation Data.....491
Pages Used for Using Custom-Defined Components to View Service Operations Data.....492
Specifying Service Operations to Associate to Custom-Defined Components.....492
Associating Service Operations to Custom-Defined Components.....492
Purging Runtime Monitor Tables.....493
Using the Services Operations Monitor Component Interface.....494
Using PeopleCode to Read and Write Errors to the Asynchronous Error Queue.....495

Chapter 22
Managing Error Handling, Logging, Tracing, and Debugging.....497
Understanding Error Handling, Logging, Tracing and Debugging.....497
Understanding Integration Gateway Error Handling.....497
Target Connector Error Handling.....497
Listening Connector Error Handling.....498
Integration Gateway Exception Types.....498
Managing Integration Gateway Message and Error Logging.....499
Understanding Message and Error Logging.....499
Setting Up Message and Error Logging.....500
Viewing Non-English Characters in Integration Gateway Log Files.....500
Managing Message Logging.....500
Managing Error Logging.....501
Managing Application Server Logging and Tracing.....503
Debugging Integrations.....503
Debugging Handler PeopleCode.....503
Handling Common Issues.....504

Copyright © 1988-2007, Oracle. All rights reserved.

Chapter 23
Providing Services.....507
Understanding Providing Services.....507
Understanding the Provide Web Service Wizard.....507
Features of the Provide Web Service Wizard.....507
Operation Types Supported.....508
Requirements for Nonrowset-Based Message Schemas.....508
Locations for Publishing WSDL Documents.....508
UDDI Repositories and Endpoints.....509
WSDL URL Formats.....509
Provided WSDL Documents.....509
PartnerLinkType Support.....518
WSDL Document Versioning.....520
Prerequisites for Providing Services.....521
Common Elements Used in This Chapter.....522
Providing Services.....522
Understanding Using the Provide Web Service Wizard.....522
Pages Used in Using the Provide Web Service Wizard.....523
Step 1: Select Services to Provide.....524
Step 2: Select Service Operations.....524
Step 3: View WSDL Documents.....525
Step 4: Specify Publishing Options.....526
Step 5: View the WSDL Generation Log.....528
Accessing Generated WSDL Documents.....529
Pages Used to Access Generated WSDL Documents.....529
Using WSDL URLs To Access Generated WSDL Documents.....529
Using the WSDL Repository to Access Generated WSDL Documents.....529
Deleting WSDL Documents.....531
Understanding Deleting WSDL Documents.....531
Page Used to Delete WSDL Documents.....532
Deleting a WSDL Document.....532

Chapter 24
Consuming Services.....533
Understanding Consuming Services.....533
Understanding the Consume Web Service Wizard.....533
Consume Web Service Wizard Features.....533
Operation Types Supported.....533
Sources for Consuming WSDL Document.....534

Copyright © 1988-2007, Oracle. All rights reserved.

Integration Metadata Created by the Consume Web Service Wizard.....534
Multiple Fault Messages.....535
Multiple Root Elements in Message Schemas.....535
Delivered Queues and Nodes.....535
Binding Style of Consumed WSDL Documents.....535
Working with Asynchronous Request/Response Service Operations.....536
Prerequisites for Consuming Services.....536
Common Elements Used in This Chapter.....536
Setting the PS_FILEDIR Environment Variable for Consuming WSDL from Files.....537
Understanding Setting the PS_FILEDIR Environment Variable.....537
Setting PS_FILEDIR in Microsoft Windows Environments.....537
Setting PS_FILEDIR in Unix Environments.....538
Using the Consume Web Service Wizard.....538
Pages Used to Consume Services.....539
Step 1: Select WSDL Source.....539
Step 2: Select Service.....541
Step 3: Select Service Ports.....541
Step 4: Select Service Operations.....542
Step 5: Convert Asynchronous Operations.....542
Step 6: Rename Operation Messages.....544
Step 7: Select a Queue for Asynchronous Operations.....546
Step 8: Select the Receiver Node.....547
Confirm and View Results.....548
Accessing Integration Metadata for Consumed Services.....549
Pages Used to Access Integration Metadata for Consumed Services.....550
Accessing Integration Metadata for a Consumed Service.....550

Chapter 25
Integrating with BPEL Process-Based Services.....553
Understanding Integrating with BPEL Processes.....553
Oracle BPEL Process Manager.....553
PeopleSoft-Delivered Application Classes for BPEL Integrations.....553
Monitoring Integrations.....554
Prerequisites for Integrating with BPEL Processes.....554
Configuring the PeopleSoft-Delivered BPEL Node.....555
Consuming BPEL Process–Based Services.....556
Understanding Consuming BPEL Process-Based Services.....556
Deploying BPEL Processes.....557
Consuming WSDL Documents from BPEL Processes.....557

Copyright © 1988-2007, Oracle. All rights reserved.

Consuming Synchronous BPEL Operations.....557
Consuming Asynchronous Request/Response BPEL Operations.....559
Consuming Asynchronous Fire-and-Forget (One-Way) BPEL Operations.....562
Providing PeopleSoft Services to BPEL Processes.....564
Understanding Providing PeopleSoft Services to BPEL Processes.....564
Providing Synchronous PeopleSoft Operations to BPEL Processes.....565
Providing Asynchronous PeopleSoft Request/Response Operations to BPEL Processes.....568

Chapter 26
Integrating with ERP Systems.....571
Understanding Integrations with ERP Systems.....571
Understanding iWay SOAPswitch.....571
Installing iWay SOAPswitch.....572
Components.....572
Delivered Adapters.....572
Operation Types.....573
Web Services and Notifications.....573
iWay SOAPswitch and Adapter Documentation.....573
Prerequisites.....574
Starting and Stopping iWay SOAPswitch.....575
Common Elements Used in This Section.....575
Starting iWay SOAPswitch Server.....575
Stopping iWay SOAPswitch Server.....575
Logging In to iWay SOAPswitch.....575
Logging In to iWay SOAPswitch.....575
Changing the iWay SOAPswitch Login User ID and Password.....576
Generating WSDL Using the ERP Connectors.....577
Configuring ERP Connectors.....578
Adding Systems.....578
Adding Consumer Systems.....578
Adding Roles and Users.....579
Creating Web Services.....579
Adding Destinations.....580
Creating Notifications.....580

Chapter 27
Setting Up Secure Integration Environments.....583
Understanding Securing Integration Environments.....583

Copyright © 1988-2007, Oracle. All rights reserved.

Web Server SSL Encryption....583
WS-Security....584
Client Authentication....584
Nonrepudiation....584
User Authentication....584
Node Authentication....585
Service Operation Permission Lists....585
Understanding PeopleSoft Integration Broker Security Processing....585
Outbound Integration Broker Security Processing....585
Inbound Integration Broker Security Processing....587
Understanding Digital Certificates....588
Digital Certificates....588
Digital Certificate Authorities....588
Digital Certificate Installation Elements....589
Installing Application Server-Based Digital Certificates....591
Understanding Installing Application Server-Based Digital Certificates....591
Pages Used to Install Application Server-Based Digital Certificates....592
Installing Application Server-Based Digital Certificates....592
Accessing Certificate Properties....596
Exporting and Converting Certificates....597
Installing Integration Gateway-Based Digital Certificates....598
Understanding Integration Gateway-Based Digital Certificates....598
Generating Private and Public Key Pairs....599
Generating CSRs....600
Obtaining Signed Root Certificates....601
Importing Signed Root Certificates....601
Specifying the Keystore Location for WS-Security....602
Encrypting Keystore Passwords for WS-Security....603
Installing Web Server-Based Digital Certificates....603
Understanding Installing Web Server-Based Digital Certificates....603
Installing Digital Certificates for SSL on BEA WebLogic....604
Installing Digital Certificates SSL Encryption on IBM WebSphere....608
Installing Digital Certificates for SSL Encryption on Oracle Application Server....612
Implementing Web Server SSL Encryption....616
Understanding Web Server SSL Encryption....616
Prerequisites for Implementing Web Server SSL Encryption....619
Configuring Web Server SSL Encryption....619
Implementing Web Server SSL Encryption....619
Implementing WS-Security....619
Understanding Implementing WS-Security in PeopleSoft Integration Broker....620

Copyright © 1988-2007, Oracle. All rights reserved.

Understanding WS-Security Processing in PeopleSoft Integration Broker.....621
Prerequisites for Implementing WS-Security in PeopleSoft Integration Broker.....624
Implementing WS-Security for Inbound Integrations.....624
Implementing WS-Security for Outbound Integrations.....624
Describing WS-Security Configuration Options for Outbound Integrations.....626
WS-Security SOAP Header Examples.....628
Implementing Client Authentication.....632
Understanding Client Authentication.....632
Implementing Nonrepudiation.....632
Understanding Nonrepudiation.....632
Prerequisites for Implementing Nonrepudiation.....637
Configuring Nonrepudiation.....637
Managing User Authentication.....637
Understanding User Authentication.....637
Understanding Outbound User Authentication.....638
Understanding Inbound User Authentication.....642
Pages Used to Manage User Authentication.....647
Activating User Authentication on Service Operations.....647
Setting Up User Authentication on Sending Systems.....647
Implementing Node Authentication.....648
Understanding Node Authentication.....648
Setting Up Password-Based Node Authentication.....648
Setting Up Certificate-Based Node Authentication.....648
Securing Service Operations with Permission Lists.....649

Chapter 28
Tuning Messaging System Performance.....651
Understanding Tuning Messaging System Performance.....651
Throttling Dispatched Messages Through the Messaging System.....651
Using Multi-Threading to Send Groups of Messages in Parallel.....652
Understanding Multi-Threading.....652
Specifying the Number of Available Threads.....652
Implementing Multi-Threading.....653
Exception Handling for Synchronous Message Processing.....654
Implementing Master-Slave Dispatchers.....656
Understanding Implementing Master-Slave Dispatchers.....656
Configuring Dynamic Slave Dispatchers.....657
Configuring Static Slave Dispatchers.....657
Configuring Integration Gateways for Load Balancing.....657

Copyright © 1988-2007, Oracle. All rights reserved.

Understanding Configuring Integration Gateways for Load Balancing.....657
Configuring Load Balancing.....658
Implementing Load Balancing on Service Operation Queues.....659
Understanding Implementing Load Balancing on Service Operation Queues.....659
Setting the Load Balance Interval Parameter.....659
Resubmitting Failed Transactions.....659
Utilize Data Mover Scripts for Large Message Subscriptions.....660

Chapter 29
Using the Inbound File Loader Utility.....661
Understanding the Inbound File Loader Utility.....661
File Processing.....661
Understanding Development Activities.....663
General Development Activities.....663
Development Activities for PeopleSoft Integration Broker Processing.....663
Creating File Layout Definitions.....664
Development Activities for Application Class Processing.....665
Prerequisites for Using the Inbound File Loader Utility.....667
Setting Up Inbound File Loader Processing Rules.....667
Understanding Setting Up Inbound File Loader Processing Rules.....668
Page Used to Set Up Inbound File Loader Processing Rules.....668
Setting Up Inbound File Loader Processing Rules.....668
Initiating File Processing.....671
Understanding Initiating File Processing.....671
Page Used to Initiate Inbound Flat File Processing.....671
Initiating Inbound Flat File Processing.....672
Testing Inbound Flat File Processing.....673

Chapter 30
Backporting Integration Metadata.....675
Understanding Backporting Integration Metadata.....675
Metadata Backported.....675
Using the Metadata Backport Utility.....676
Page Used to Backport Integration Metadata.....676
Backporting Integration Metadata.....676
Working with Backported Projects.....677
Cleaning Up PeopleTools 8.49 Databases After Backporting Metadata.....677

Copyright © 1988-2007, Oracle. All rights reserved.

Appendix A
Integration Scenarios.....679
Understanding Integration Setup.....679
Integrating with PeopleSoft Integration Broker Systems.....683
Understanding This Scenario.....683
Configuring the System for This Scenario.....684
Integrating with PeopleSoft Integration Broker Systems Through Firewalls.....685
Understanding This Scenario.....686
Configuring the System for This Scenario.....687
Integrating with PeopleSoft Integration Broker Systems by Using Hubs.....688
Understanding This Scenario.....689
Understanding Hub Routing Types.....689
Configuring Generic-Routing Hubs.....690
Configuring Sender-Specified Routing Hubs.....692
Integrating with Third-Party Systems.....695
Understanding This Scenario.....695
Configuring the System for This Scenario.....696
Integrating with Third-Party Systems by Using Remote Gateways.....697
Understanding This Scenario.....697
Sending Messages to Third-Party Systems.....698
Receiving Messages from Third-Party Systems.....699
Integrating with PeopleSoft 8.1x Systems.....702
Understanding This Scenario.....702
Configuring the System for This Scenario.....703

Appendix B
Using the Delivered Listening Connectors and Target Connectors.....705
Understanding Using This Appendix.....705
Prerequisites.....705
Setting Up Metadata.....706
Understanding Setting Up Metadata.....706
Prerequisites.....706
Creating Services, Service Operations, Queues, and Messages.....706
Creating the Test Record and Page.....707
Creating Nodes and Routing Definitions.....708
Setting Up Integration Gateway Logging.....709
Example 1: Using the PeopleSoft Connectors.....709
Understanding the PeopleSoft Connector Examples.....709
Prerequisites.....709

Copyright © 1988-2007, Oracle. All rights reserved.

Using the PeopleSoft Target Connector....709
Using the PeopleSoft Listening Connector....711
Example 2: Using the HTTP Connectors....713
Prerequisites....713
Using the HTTP Listening Connector....713
Using the HTTP Target Connector....717
Example 3: Using the PeopleSoft 8.1 Connectors....718
Understanding the PeopleSoft 8.1 Connectors Examples....718
Setting Up Data for the PeopleSoft 8.1 Connectors Examples....718
Using the PeopleSoft 8.1 Target Connector....721
Using the PeopleSoft 8.1 Listening Connector....721
Example 4: Using the JMS Connectors....721
Understanding the JMS Connector Examples....721
Prerequisites....722
Using the JMS Target Connector....722
Using the JMS Listening Connector....723
Example 5: Using the AS2 Connectors....725
Understanding the AS2 Connector Examples....725
Prerequisites....725
Using the AS2 Target Connector....725
Using the AS2 Listening Connector....727
Example 6: Using the Simple File Target Connector....729
Writing PeopleSoft Data to Files....729
Reading Data Into PeopleSoft From Files....730
Example 7: Using the FTP Target Connector....731
Prerequisites....731
Uploading Files to FTP Servers....732
Downloading Files From FTP Servers....732
Example 8: Using the SMTP Target Connector....734
Prerequisites....734
Sending Email Messages to SMTP Servers....734

Appendix C
Transformation Example: Integration Between Two PeopleSoft Nodes....737
Understanding This Appendix....737
Using the Example....737
Integration Metadata for This Example....737
Creating Message Definitions....738
Message Definition: PeopleSoft SCM Node....738

Copyright © 1988-2007, Oracle. All rights reserved.

Message Definition: PeopleSoft CRM Node.....739
Setting Up the Codesets.....741
Setting Up the Transformation.....742
XSL Walkthrough.....745
Transformation Processing: First Pass.....745
Transformation Processing: Second Pass.....748
Testing the Transformation.....749

Appendix D
Using the Integration Broker Connector SDK.....751
Understanding the PeopleSoft Integration Broker Connector SDK.....751
The PeopleSoft Integration Broker Connector SDK.....751
SDK Contents.....752
SDK Location.....752
SDK API Documentation.....752
Developing Connectors.....753
Understanding Connector Development and Implementation.....753
Understanding General Connector Class Development Considerations.....756
Developing Target Connector Classes.....757
Developing Listening Connector Classes.....761
Installing Connector Classes.....765
Registering Connectors.....765
Using Connector Templates.....766
Developing Connectors Based on DOM.....770
Understanding the Java XML DOM Wrapper.....770
Using Java XML DOM Wrapper Classes.....770
Developing and Implementing Connectors in the C/C++ Environment.....773
Understanding the Development Process.....773
Creating Target Connectors for the C/C++ Environment.....774
Reusing Connector Code.....777
Reusing Connector Code Through Inheritance.....777
Reusing Connector Code Through Delegation.....777

Appendix E
Understanding Migrated Integration Metadata.....779
Understanding Migrated Integration Metadata.....779
Node Objects.....779
Channel Objects.....779

Copyright © 1988-2007, Oracle. All rights reserved.

Message Objects.....780
Node Transaction and Relationship Objects.....780
Understanding Migrated Integration PeopleCode.....781
Application Classes.....781
PeopleCode Methods.....782
Built-In Functions.....782
Other Migrated Constructs.....782
Special Characters.....783
Correcting Integration PeopleCode That Did Not Migrate.....783
Understanding Integration PeopleCode That Did Not Migrate.....783
Correcting Non-Migrated Integration PeopleCode.....783

Appendix F
Data Dependencies and Relationships When Copying Data Between Databases.....787
Understanding Data Dependencies and Relationships for Copying Data.....787
Using Data Mover Scripts to Move Non-Managed Object Data.....789

Appendix G
Technologies, Specifications, and Products Supported by PeopleSoft Integration Broker.....791
Supported technologies, specifications and Third-Party Products.....791

Glossary of PeopleSoft Enterprise Terms.....793

Index 819

Copyright © 1988-2007, Oracle. All rights reserved.

Copyright © 1988-2007, Oracle. All rights reserved.

About This PeopleBook

PeopleSoft Enterprise PeopleBooks provide you with the information that you need to implement and use PeopleSoft Enterprise applications from Oracle.

This preface discusses:

- PeopleSoft Enterprise application prerequisites.
- Application fundamentals.
- Documentation updates and printed documentation.
- Additional resources.
- Typographical conventions and visual cues.
- Comments and suggestions.
- Common elements in PeopleBooks.

Note. PeopleBooks document only elements, such as fields and check boxes, that require additional explanation. If an element is not documented with the process or task in which it is used, then either it requires no additional explanation or it is documented with common elements for the section, chapter, PeopleBook, or product line. Elements that are common to all PeopleSoft Enterprise applications are defined in this preface.

PeopleSoft Enterprise Application Prerequisites

To benefit fully from the information that is covered in these books, you should have a basic understanding of how to use PeopleSoft Enterprise applications.

You might also want to complete at least one introductory training course, if applicable.

You should be familiar with navigating the system and adding, updating, and deleting information by using PeopleSoft Enterprise menus, pages, or windows. You should also be comfortable using the World Wide Web and the Microsoft Windows or Windows NT graphical user interface.

These books do not review navigation and other basics. They present the information that you need to use the system and implement your PeopleSoft Enterprise applications most effectively.

Application Fundamentals

Each application PeopleBook provides implementation and processing information for your PeopleSoft Enterprise applications.

For some applications, additional, essential information describing the setup and design of your system appears in a companion volume of documentation called the application fundamentals PeopleBook. Most product lines have a version of the application fundamentals PeopleBook. The preface of each PeopleBook identifies the application fundamentals PeopleBooks that are associated with that PeopleBook.

Copyright © 1988-2007, Oracle. All rights reserved.

The application fundamentals PeopleBook consists of important topics that apply to many or all PeopleSoft Enterprise applications. Whether you are implementing a single application, some combination of applications within the product line, or the entire product line, you should be familiar with the contents of the appropriate application fundamentals PeopleBooks. They provide the starting points for fundamental implementation tasks.

Documentation Updates and Printed Documentation

This section discusses how to:

- Obtain documentation updates.
- Download and order printed documentation.

Obtaining Documentation Updates

You can find updates and additional documentation for this release, as well as previous releases, on Oracle's PeopleSoft Customer Connection website. Through the Documentation section of Oracle's PeopleSoft Customer Connection, you can download files to add to your PeopleBooks Library. You'll find a variety of useful and timely materials, including updates to the full line of PeopleSoft Enterprise documentation that is delivered on your PeopleBooks CD-ROM.

Important! Before you upgrade, you must check Oracle's PeopleSoft Customer Connection for updates to the upgrade instructions. Oracle continually posts updates as the upgrade process is refined.

See Also

Oracle's PeopleSoft Customer Connection, http://www.oracle.com/support/support_peoplesoft.html

Downloading and Ordering Printed Documentation

In addition to the complete line of documentation that is delivered on your PeopleBook CD-ROM, Oracle makes PeopleSoft Enterprise documentation available to you via Oracle's website. You can:

- Download PDF files.
- Order printed, bound volumes.

Downloading PDF Files

You can download PDF versions of PeopleSoft Enterprise documentation online via the Oracle Technology Network. Oracle makes these PDF files available online for each major release shortly after the software is shipped.

See Oracle Technology Network, http://www.oracle.com/technology/documentation/psftent.html.

Ordering Printed, Bound Volumes

You can order printed, bound volumes of selected documentation via the Oracle Store.

See Oracle Store, http://oraclestore.oracle.com/OA_HTML/ibeCCtpSctDspRte.jsp?section=14021

Copyright © 1988-2007, Oracle. All rights reserved.

Additional Resources

The following resources are located on Oracle's PeopleSoft Customer Connection website:

Resource	Navigation
Application maintenance information	Updates + Fixes
Business process diagrams	Support, Documentation, Business Process Maps
Interactive Services Repository	Support, Documentation, Interactive Services Repository
Hardware and software requirements	Implement, Optimize + Upgrade; Implementation Guide; Implementation Documentation and Software; Hardware and Software Requirements
Installation guides	Implement, Optimize + Upgrade; Implementation Guide; Implementation Documentation and Software; Installation Guides and Notes
Integration information	Implement, Optimize + Upgrade; Implementation Guide; Implementation Documentation and Software; Pre-Built Integrations for PeopleSoft Enterprise and JD Edwards EnterpriseOne Applications
Minimum technical requirements (MTRs)	Implement, Optimize + Upgrade; Implementation Guide; Supported Platforms
Documentation updates	Support, Documentation, Documentation Updates
PeopleBooks support policy	Support, Support Policy
Prerelease notes	Support, Documentation, Documentation Updates, Category, Release Notes
Product release roadmap	Support, Roadmaps + Schedules
Release notes	Support, Documentation, Documentation Updates, Category, Release Notes
Release value proposition	Support, Documentation, Documentation Updates, Category, Release Value Proposition
Statement of direction	Support, Documentation, Documentation Updates, Category, Statement of Direction
Troubleshooting information	Support, Troubleshooting
Upgrade documentation	Support, Documentation, Upgrade Documentation and Scripts

Copyright © 1988-2007, Oracle. All rights reserved.

Typographical Conventions and Visual Cues

This section discusses:

- Typographical conventions.
- Visual cues.
- Country, region, and industry identifiers.
- Currency codes.

Typographical Conventions

This table contains the typographical conventions that are used in PeopleBooks:

Typographical Convention or Visual Cue	Description
Bold	Indicates PeopleCode function names, business function names, event names, system function names, method names, language constructs, and PeopleCode reserved words that must be included literally in the function call.
Italics	Indicates field values, emphasis, and PeopleSoft Enterprise or other book-length publication titles. In PeopleCode syntax, italic items are placeholders for arguments that your program must supply. We also use italics when we refer to words as words or letters as letters, as in the following: Enter the letter *O*.
KEY+KEY	Indicates a key combination action. For example, a plus sign (+) between keys means that you must hold down the first key while you press the second key. For ALT+W, hold down the ALT key while you press the W key.
`Monospace font`	Indicates a PeopleCode program or other code example.
“ ” (quotation marks)	Indicate chapter titles in cross-references and words that are used differently from their intended meanings.
. . . (ellipses)	Indicate that the preceding item or series can be repeated any number of times in PeopleCode syntax.
{ } (curly braces)	Indicate a choice between two options in PeopleCode syntax. Options are separated by a pipe (|).

Copyright © 1988-2007, Oracle. All rights reserved.

Typographical Convention or Visual Cue	Description
[ ] (square brackets)	Indicate optional items in PeopleCode syntax.
& (ampersand)	When placed before a parameter in PeopleCode syntax, an ampersand indicates that the parameter is an already instantiated object. Ampersands also precede all PeopleCode variables.

Visual Cues

PeopleBooks contain the following visual cues.

Notes

Notes indicate information that you should pay particular attention to as you work with the PeopleSoft Enterprise system.

Note. Example of a note.

If the note is preceded by *Important!,* the note is crucial and includes information that concerns what you must do for the system to function properly.

Important! Example of an important note.

Warnings

Warnings indicate crucial configuration considerations. Pay close attention to warning messages.

Warning! Example of a warning.

Cross-References

PeopleBooks provide cross-references either under the heading "See Also" or on a separate line preceded by the word *See.* Cross-references lead to other documentation that is pertinent to the immediately preceding documentation.

Country, Region, and Industry Identifiers

Information that applies only to a specific country, region, or industry is preceded by a standard identifier in parentheses. This identifier typically appears at the beginning of a section heading, but it may also appear at the beginning of a note or other text.

Example of a country-specific heading: "(FRA) Hiring an Employee"

Example of a region-specific heading: "(Latin America) Setting Up Depreciation"

Country Identifiers

Countries are identified with the International Organization for Standardization (ISO) country code.

Copyright © 1988-2007, Oracle. All rights reserved.

Region Identifiers

Regions are identified by the region name. The following region identifiers may appear in PeopleBooks:

- Asia Pacific
- Europe
- Latin America
- North America

Industry Identifiers

Industries are identified by the industry name or by an abbreviation for that industry. The following industry identifiers may appear in PeopleBooks:

- USF (U.S. Federal)
- E&G (Education and Government)

Currency Codes

Monetary amounts are identified by the ISO currency code.

Comments and Suggestions

Your comments are important to us. We encourage you to tell us what you like, or what you would like to see changed about PeopleBooks and other Oracle reference and training materials. Please send your suggestions to your product line documentation manager at Oracle Corporation, 500 Oracle Parkway, Redwood Shores, CA 94065, U.S.A. Or email us at appsdoc@us.oracle.com.

While we cannot guarantee to answer every email message, we will pay careful attention to your comments and suggestions.

Common Elements Used in PeopleBooks

As of Date	The last date for which a report or process includes data.
Business Unit	An ID that represents a high-level organization of business information. You can use a business unit to define regional or departmental units within a larger organization.
Description	Enter up to 30 characters of text.
Effective Date	The date on which a table row becomes effective; the date that an action begins. For example, to close out a ledger on June 30, the effective date for the ledger closing would be July 1. This date also determines when you can view and change the information. Pages or panels and batch processes that use the information use the current row.

Copyright © 1988-2007, Oracle. All rights reserved.

Once, Always, and **Don't Run**	Select Once to run the request the next time the batch process runs. After the batch process runs, the process frequency is automatically set to Don't Run. Select Always to run the request every time the batch process runs. Select Don't Run to ignore the request when the batch process runs.
Process Monitor	Click to access the Process List page, where you can view the status of submitted process requests.
Report Manager	Click to access the Report List page, where you can view report content, check the status of a report, and see content detail messages (which show you a description of the report and the distribution list).
Request ID	An ID that represents a set of selection criteria for a report or process.
Run	Click to access the Process Scheduler request page, where you can specify the location where a process or job runs and the process output format.
SetID	An ID that represents a set of control table information, or TableSets. TableSets enable you to share control table information and processing options among business units. The goal is to minimize redundant data and system maintenance tasks. When you assign a setID to a record group in a business unit, you indicate that all of the tables in the record group are shared between that business unit and any other business unit that also assigns that setID to that record group. For example, you can define a group of common job codes that are shared between several business units. Each business unit that shares the job codes is assigned the same setID for that record group.
Short Description	Enter up to 15 characters of text.
User ID	An ID that represents the person who generates a transaction.

Copyright © 1988-2007, Oracle. All rights reserved.

Copyright © 1988-2007, Oracle. All rights reserved.

PeopleSoft Integration Broker Preface

This preface provides an overview of the PeopleSoft Integration Broker PeopleBook.

PeopleSoft Integration Broker

PeopleSoft Integration Broker facilitates exposing PeopleSoft business logic as services and consuming external web services for PeopleSoft applications to invoke. PeopleSoft Integration Broker also supports synchronous and asynchronous messaging with other PeopleSoft applications and with third-party systems. PeopleSoft Integration Broker uses a variety of communication protocols, while managing message structure, message content, and transport disparities.

This PeopleBook describes the processes for using PeopleSoft Integration Broker to develop and administer web services. These processes include configuring messaging systems and managing integration gateways. They also include defining services and service operations, messages, queues, nodes, routings, and transformations.

This PeopleBook also discusses developing the necessary PeopleCode to send, receive, and route service operations, as well as developing PeopleCode and XSLT code to filter, transform and translate message content.

This PeopleBook also describes monitoring service operation activity, managing errors, and debugging code, as well as developing listening and target connectors to accommodate additional communication protocols.

Copyright © 1988-2007, Oracle. All rights reserved.

Copyright © 1988-2007, Oracle. All rights reserved.

CHAPTER 1

Getting Started with PeopleSoft Integration Broker

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

- Plan the integration architecture.
- Plan integrations.
- Determine security.
- Plan for support.
- Assess staff skills.

PeopleSoft Integration Broker Overview

This PeopleBook describes using PeopleSoft Integration Broker to

- Perform asynchronous and synchronous messaging among internal systems and third-party systems.
- Expose PeopleSoft business logic as web services to PeopleSoft and third-party systems
- Consume and invoke web services from third-party and PeopleSoft systems

Implementing 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.

Copyright © 1988-2007, Oracle. All rights reserved.

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.

Planning Integrations

In planning the integrations to develop and execute, consider the following:

- Real-time integrations or scheduled integrations.

  Determine if you business needs are best served with real-time integration or scheduled integrations.

  Scheduled batch processing and file loads are discussed in other PeopleBooks.

  See *Enterprise PeopleTools 8.49 PeopleBook: PeopleSoft Process Scheduler*, "Getting Started With PeopleSoft Process Scheduler" and *Enterprise PeopleTools 8.49 PeopleBook: PeopleSoft Application Engine*, "Getting Started With PeopleSoft Application Engine".

- Inventory the integrations to develop.

  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?

- Generic integrations.

  Can you develop generic integrations? Perhaps in your current environment only two systems need to exchange information and they can do so in a proprietary way. But consider that one day perhaps additional systems in your enterprise may also need to exchange that information with the source system. Will you need to develop transformations for system that will be integrating later on? Can you develop the integration in a way so that other systems will be able to consume the service or subscribe to the information without requiring complex transformations?

- Determine which integrations will require synchronous messaging and which ones will require asynchronous messaging.

  In PeopleSoft Integration Broker synchronous integration, all processing stops until a response is received. In PeopleSoft Integration Broker asynchronous integration, each request is placed in a queue to be processed as soon as the system can accommodate the request. Perhaps you may need to stop processing of fulfilling an order until the system verifies if all requested items are available in inventory. Processing of all support tickets probably should not stop if a system uses integration to add a new ticket to a queue.

- Prioritize integration development.

  Plan to develop mission critical integrations first, standard integrations next, and nice-to-have integrations last.

- Determine if data will need transformation or translation.

- Plan on using integration simulation tools.

  Plan on using simulation tools such as PeopleSoft Send Master to simulate integration with external systems that are not under your control. Even when you do control all systems that are being integrated, if you can't get the integration to work using Send Master, you definitely won't be able to get it working from the external system. Test integrations using Send Master before spending hours debugging a system.

 Copyright © 1988-2007, Oracle. All rights reserved.

See *Enterprise PeopleTools 8.49 PeopleBook: PeopleSoft Integration Testing Utilities and Tools*, "Getting Started with PeopleSoft Integration Testing Utilities and Tools".

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.

Planning for Support

Develop a support plan for after "go-live." In doing so, consider the following:

- Determine who in your organization will support integration development and administration.
- Determine the type of error-notification and exception handling to implement to meet your support requirements. Consider that while system administrators can resolve communication failure between machines, they may not be able to resolve errors resulting from one system transmitting bad data to another. Analyst intervention may be required to correct the data. Stronger validation at point of data entry will result in fewer calls to a functional analyst to resolve integration issues.

Assessing Staff Skills

Assess the skills of the people who will perform development and administrative functions.

Developers working on the implementation of PeopleSoft Integration Broker should have familiarity, training or experience in the following PeopleSoft areas:

- PeopleTools.
- PeopleCode.
- Application Engine.

In addition, developers should have an understanding and research capabilities in:

- Extensible Markup Language (XML).
- XML schema.
- Simple Object Access Protocol (SOAP).
- Hypertext Transfer Protocol (HTTP).
- Web Services Description Language (WSDL).
- Universal Description, Discovery and Integration (UDDI) standard.
- Java programming language.

Copyright © 1988-2007, Oracle. All rights reserved.

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.

Other Sources of Information

In addition to the implementation considerations presented in this chapter, take advantage of all PeopleSoft sources of information, including the installation guides, release notes, PeopleBooks, curriculum and red papers.

See Also

"PeopleSoft Integration Broker Preface," page xliii

Enterprise PeopleTools 8.49 PeopleBook: Getting Started with PeopleTools

 Copyright © 1988-2007, Oracle. All rights reserved.

CHAPTER 2

Understanding PeopleSoft Integration Broker

This chapter provides overview information about PeopleSoft Integration Broker and discusses:

- Integration gateway architecture.
- Integration engine architecture.
- Transaction types.
- Incoming and outgoing message flows.

Important! PeopleSoft Integration Broker interacts with a wide variety of third-party products. This PeopleBook is not an authoritative source of information about any third-party product. Most third-party products are delivered with their own documentation, which you should use as the primary source for information about them. This PeopleBook provides guidance that enables you to determine the configuration settings that PeopleSoft Integration Broker requires to work with third-party products. It does not address all configuration permutations. Examples of settings and data relative to a third-party product may not be correct for your particular situation. To properly configure PeopleSoft Integration Broker, you must apply your own expertise and obtain the most accurate and current information about third-party products.

Introduction to PeopleSoft Integration Broker

PeopleSoft Integration Broker is a middleware technology that:

- Performs asynchronous and synchronous messaging among internal systems and third-party systems.
- Exposes PeopleSoft business logic as web services to PeopleSoft and third-party systems.
- Consumes and invokes web services from third-party and PeopleSoft systems.

PeopleSoft Integration Broker enables you to perform these integrations among internal systems and third-party integration partners, while managing data structure, data format and transport disparities. Because of its modular design, you can reuse many elements that you develop for integrations.

PeopleSoft Integration Broker consists of two subsystems: the integration gateway and the integration engine. The integration gateway resides on a PeopleSoft web server, and the integration engine is installed on an application server as part of the PeopleSoft application.

Web Services

PeopleSoft Integration Broker enables you to provide web services to other PeopleSoft systems and external integration partners by generating Web Services Description Language (WSDL) documents from integration metadata. PeopleSoft supports providing WSDL documents to the PeopleSoft WSDL repository and Universal Description, Discovery, and Integration (UDDI) repositories. Service introspection from Web Service Inspection Language (WSIL) documents is also supported.

Copyright © 1988-2007, Oracle. All rights reserved.

The system enables you to consume WSDL documents from other PeopleSoft systems and third-party systems, and automatically creates integration metadata based on the consumed WSDL documents for processing integrations. You can consume WSDL documents from other PeopleSoft systems, UDDI repositories, WSDL URLs, and Web Services Inspection Language (WSIL) URLs.

Integration Gateway

The integration gateway is a platform that manages the receipt and delivery of messages passed among systems through PeopleSoft Integration Broker. It supports the leading TCP/IP protocols used in the marketplace today and provides extensible interfaces to develop new connectors for communication with legacy, enterprise resource planning, and internet-based systems.

Additional features include:

- Backward compatibility for Extensible Markup Language (XML) links and PeopleSoft Application Messaging.
- Listening connectors and target connectors that transport messages between integration participants and the integration engine.

  Note. This feature also enables you to build your own connectors to complement those delivered with PeopleSoft Integration Broker.

- Basic logging information concerning message receipt, delivery, and errors.
- Connection persistence with continuous open feeds to external systems through connectors, with full failover capabilities.
- Transport protocol and message format management so that when messages reach the integration engine, they have a PeopleSoft-compatible message format.

See Also

Chapter 2, “Understanding PeopleSoft Integration Broker,” Integration Gateway Architecture, page 7

Integration Engine

The integration engine runs on the PeopleSoft application server. It’s tied closely to the PeopleSoft application, and it sends or receives messages for the application. Rather than communicating directly with other applications, the integration engine sends and receives messages through one or more separately installed integration gateways.

The integration engine:

- Uses a modular architecture, so it can treat gateways as black boxes and communicate with them using standard connectors.
- Adapts elements of an existing integration to produce a new integration with only minor adjustments.
- Handles messages containing data in a variety of formats. Formats include PeopleSoft rowset-based message format, and nonrowset-based message structures including , XML document object model messages, Simple Object Access Protocol (SOAP) messages, and non-XML files.
- Sends and receives messages asynchronously (like email) or synchronously (suspending activity to wait for a response).
- Applies message transmission type and routing based on specifications that you define in a PeopleSoft Pure Internet Architecture component.

 Copyright © 1988-2007, Oracle. All rights reserved.

- By applying application engine transform programs, transforms message structure and translates data content according to specifications that you define in PeopleSoft Pure Internet Architecture components and apply with Extensible Stylesheet Language Transformation (XSLT) code or PeopleCode.

  These specifications can be reused for other integrations.

- Handles security features such as authentication, nonrepudiation, and cookies.

See Also

Chapter 2, "Understanding PeopleSoft Integration Broker," Integration Engine Architecture, page 10

Integration Gateway Architecture

This section discusses:

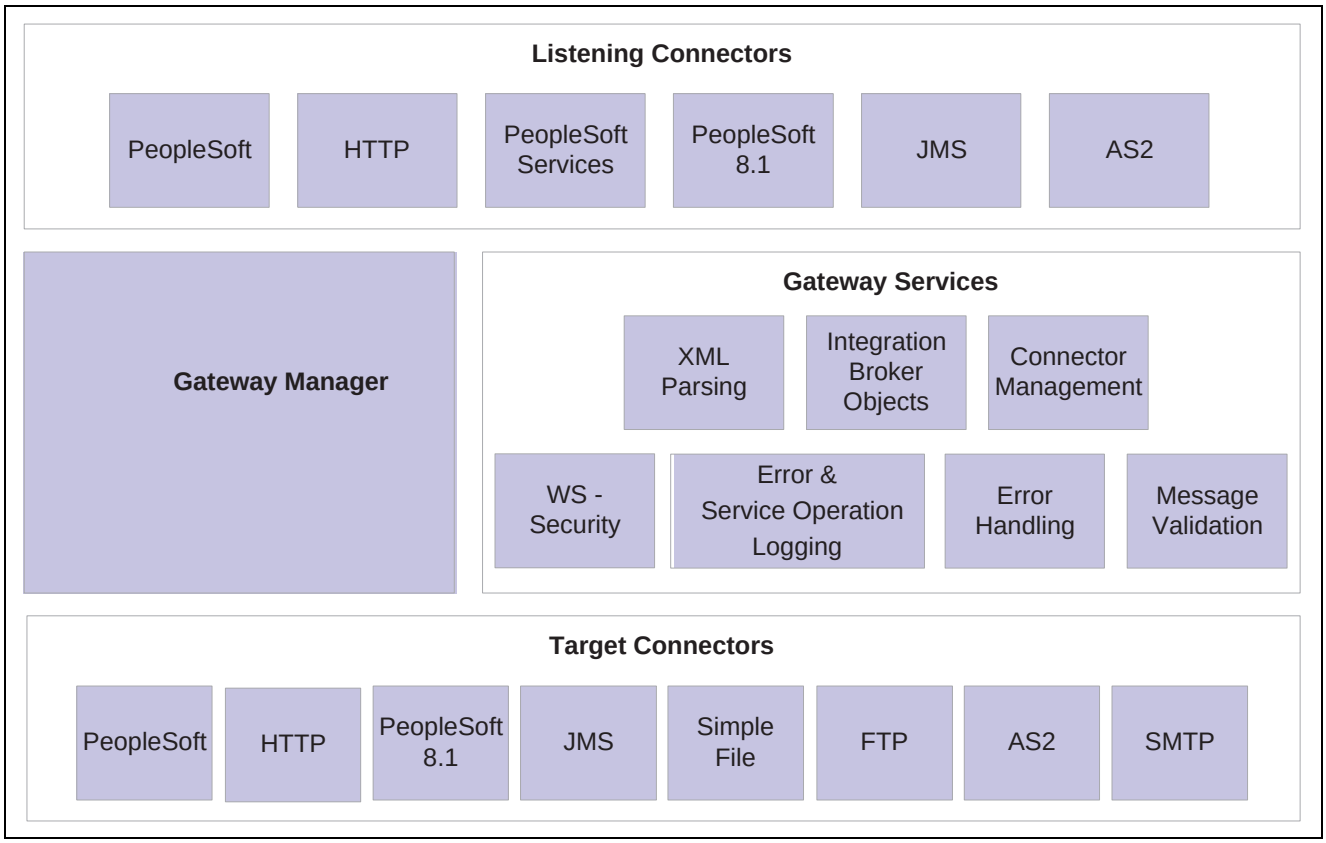
- Architecture components.
- Connectors.
- Gateway manager.
- Gateway services.

Architecture Elements

You use an integration gateway to receive and send messages among integration participant systems.

Listening connectors receive incoming messages and deliver the incoming requests to the gateway manager, which is a dispatcher for messages that flow through an integration gateway. The gateway manager determines which target connector to use to properly deliver the messages to their intended recipients. The target connector then delivers the messages to the intended recipients using the recipients' preferred protocols.

Copyright © 1988-2007, Oracle. All rights reserved.



Integration gateway architecture

Connectors

Listening connectors and target connectors transport messages between integration participants and the integration gateway. These connectors support asynchronous and synchronous message handling. Many connectors are configurable at the integration gateway and system levels.

Listening Connectors

Listening connectors receive incoming data streams and perform services based on the content of the stream. They are invoked externally by other PeopleSoft systems and third-party systems.

Target Connectors

Target connectors initiate communication with other PeopleSoft systems or third-party systems. A target connector might not receive a response from the target system during each operation, but every transmission requires a low-level acknowledgment.

PeopleSoft Integration Broker Connector SDK

The integration gateway provides a fully extensible model for developing new connectors built to the interface specification of the PeopleSoft Integration Broker software development kit (SDK) by PeopleSoft customers, consultants, and application developers.

See Also

Chapter 9, “Using Listening Connectors and Target Connectors,” page 115

Appendix D, “Using the Integration Broker Connector SDK,” page 751

 Copyright © 1988-2007, Oracle. All rights reserved.

Gateway Manager

The gateway manager processes every message that flows through an integration gateway and maintains links to the other major integration gateway components, including target connectors, listening connectors, and each of the gateway services.

Listening connectors invoke the gateway manager when they receive a request. The gateway manager uses the messaging objects IBRequest and IBResponse to determine how to route each request.

The gateway manager uses a number of the gateway services during this stage to perform operations such as message validation. The gateway manager then invokes the appropriate target connector based on the content of the message object and waits for a reply from the target connector. When the reply is received, the gateway manager forwards the reply to the calling listening connector.

If an error occurs, the gateway manager uses the error handling service and works with the service to prepare an error reply for the listening connector.

Gateway Services

This section describes the gateway services that the gateway manager uses.

Error Handling

The integration gateway provides a standard error handling interface that is exposed to each connector. This service provides error handling and error logging for most connectors delivered with PeopleSoft Integration Broker.

Integration Broker Objects

Two objects comprise the messaging objects service in the integration gateway:

- IBRequest
- IBResponse

These objects represent the request and response that enter and exit PeopleSoft Integration Broker.

See Chapter 8, “Understanding Supported Message Structures,” page 81.

XML Parsing

Most IBRequest objects and IBResponse objects that are processed in the system contain a content section that represents the actual business content sent.

Most of the time, these content sections contain XML data. Consequently, often connectors must parse and traverse XML. The standard Java XML objects are cumbersome for manipulating XML, so the integration gateway includes an XML parsing service consisting of objects that provide an intuitive interface for manipulating XML objects. This service is delivered as a set of three classes: XmlDocument, XmlNode and XmlNodeList.

See *Enterprise PeopleTools 8.49 PeopleBook: PeopleCode API Reference*.

Message Validation

Messages that pass into PeopleSoft Integration Broker must contain certain elements to be processed. Because the integration gateway is the first component that processes messages sent to a PeopleSoft application, it performs basic validation—such as making sure that the message identifies its requestor and service operation name—to ensure that the integration engine and the target application can process them.

Copyright © 1988-2007, Oracle. All rights reserved.

Connector Management

The connector management service is a composite of several services that manage connectors. The gateway processes each IBRequest to determine the appropriate connector to call in each situation. This is primarily a message routing function that has varying levels of complexity abstracted from the connectors. The connector management service also processes the IBResponse returned by each connector.

Error and Message Logging

Most components in the system use a standard error logging interface.

Each PeopleSoft-delivered connector uses the logging API in the same fashion, ensuring that an administrator can quickly drill down on problems or simply review the logs to see the IBRequest object, the IBResponse object, and even the raw data exchanged with integration participants.

See Chapter 22, “Managing Error Handling, Logging, Tracing, and Debugging,” page 497.

Integration Engine Architecture

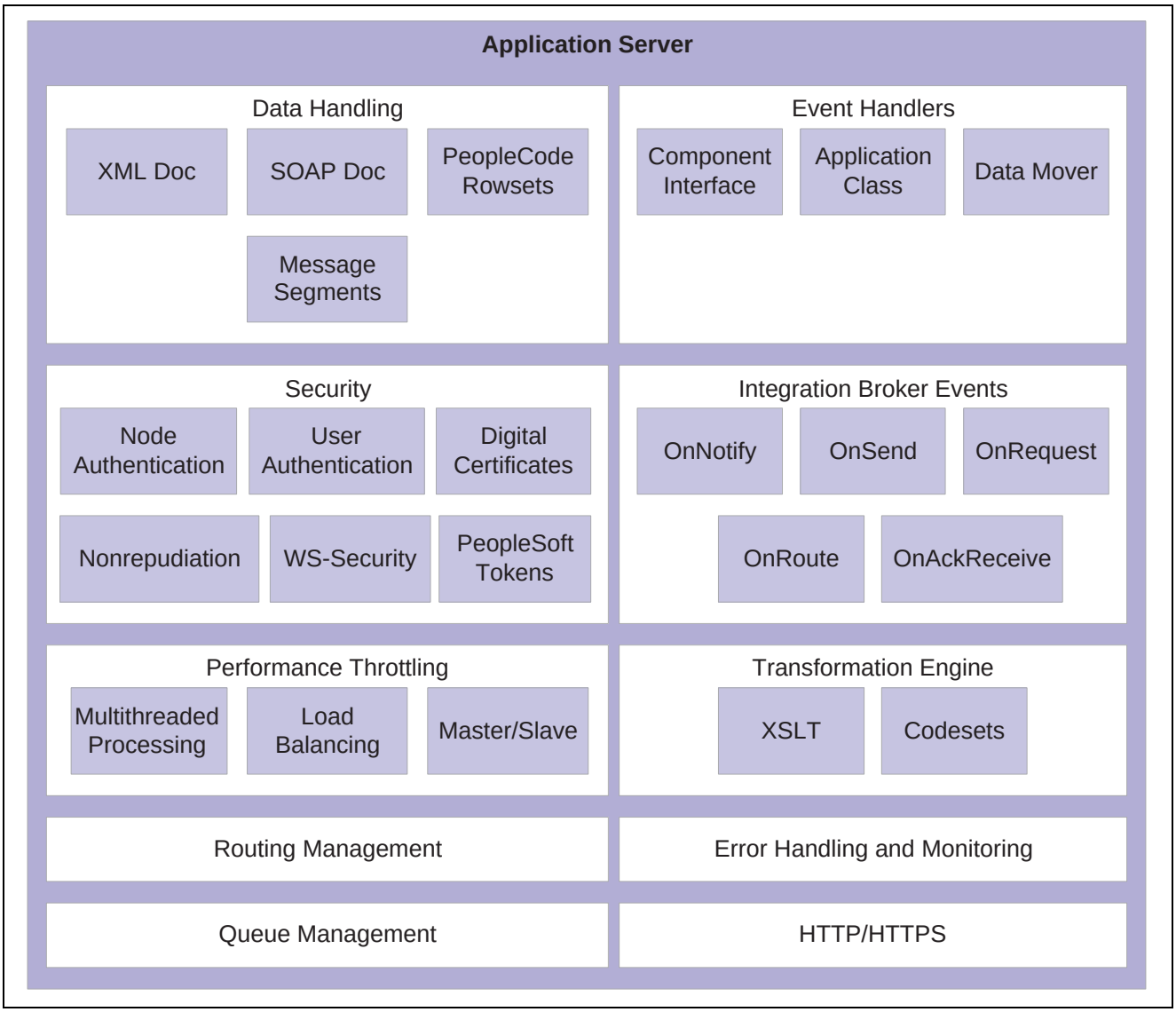
The integration engine uses a variety of PeopleTools elements to create, implement, manage, and enhance integrations. Its modular architecture separates integration development activities from administrative activities.

The integration engine is a combination of PeopleSoft Application Designer definitions, PeopleSoft Pure Internet Architecture definitions, PeopleCode, and XSLT code, along with the underlying mechanisms that tie all these elements together. The underlying mechanisms include the request handlers that process both inbound and outbound messages according to the specifications in the development and administrative elements.

The integration engine resides on the PeopleSoft application server.

The following diagram shows the integration components that reside on the integration engine and the types of processing it performs.

 Copyright © 1988-2007, Oracle. All rights reserved.



Integration engine architecture

Service Operations and Messages

A service operation definition contains the processing logic for an integration and determines how the integration is to be processed (synchronously or asynchronously). Routings specify the direction of the integration (inbound or outbound), allow you to define aliases and physical transformations and override target specifications.

Beginning with the PeopleTools 8.48 release, service operations and routings replace transactions and relationships from earlier PeopleSoft Integration Broker 8.4x versions.

You create messages and generate XML schemas from them to include in the service operation definition. The XML message schemas provides the physical description of the data that is being sent, including fields, field types, and field lengths.

Copyright © 1988-2007, Oracle. All rights reserved.

Note. In this release of PeopleSoft Integration Broker, messages contain no processing logic. All processing logic is defined using handlers. Handlers are specified in service operation definitions.

Service Operation Types

PeopleSoft Integration Broker supports four types of service operation types:

- Asynchronous one-way.
- Asynchronous request/response.
- Asynchronous to synchronous.
- Synchronous

Note. In this release of PeopleSoft Integration Broker, the term transaction is used to describe the exchange of data between integration partners.

When PeopleSoft Integration Broker sends a service operation, the receiving system returns a response back to the sender. With asynchronous transactions, the response is automatically generated by the integration gateway, and it serves only to notify the sending system of the transmission status of the request . It is processed automatically by the application server, which uses that status information to update the Service Operations Monitor. With synchronous transactions, however, the response includes the content that is requested by the sending system, and it must be generated and returned by the receiving system.

Operation Types

PeopleSoft Integration Broker supports the following operation types.

For any operation type, the application must invoke PeopleCode, a component interface or data mover script to generate and send a service operation, or to receive and process a service operation.

Operation Type	Routing	Actions
Asynchronous — One Way	Outbound.	1. The application generates and sends a request. 2. One or more target system receives and processes the request.
Synchronous	Outbound.	1. The application generates and sends a request. 2. The application suspends activity and waits for a response. 3. A single target system receives and processes the request, then generates and sends a response. 4. The application resumes its activity and receives and processes the response.

 Copyright © 1988-2007, Oracle. All rights reserved.

Operation Type	Routing	Actions
Asynchronous — Request/Response	Outbound.	1. The application generates and sends a request. 2. The target system receives and process the request. 3. Sometime later the target system sends a response (which contains the transaction ID from the original request. This ID serves as the correlation ID. 4. The application processes the response using the correlation ID to map it back to the original request.
Asynchronous to Synchronous.	Outbound.	1. The application generates and sends a request. 2. A single target system receives and processes the request, then generates and sends a response. 3. The application receives and processes the response.
Asynchronous — One way.	Inbound.	1. A source system generates and sends a request. 2. The application receives and processes the request.
Synchronous.	Inbound.	1. A source system generates and sends a request. 2. The source system suspends activity and waits for a response. 3. The application receives and processes the request, then generates and sends a response. 4. The source system resumes its activity and receives and processes the response.

Copyright © 1988-2007, Oracle. All rights reserved.

Operation Type	Routing	Actions
Asynchronous — Request/Response.	Inbound.	1. A source system generates and sends a request. 2. The application receives and processes the request. 3. Sometime later the application sends a response back to the source system. The response includes a unique identifier from the original request, which serves as a correlation ID. 4. The source system processes the response using the correlation ID to map it back to the original request.
Asynchronous to Synchronous.	Inbound.	1. A source system generates and sends a request. 2. The application receives and processes the request, then generates and sends a response. 3. The source system receives and processes the response.

See Also

Chapter 15, “Managing Service Operations,” Service Operation Types, page 291

Incoming and Outgoing Request Flows

This section discusses how incoming and outgoing service operations flow through the architecture components of PeopleSoft Integration Broker.

The PeopleSoft messaging architecture is discussed in greater detail in the Understanding Messaging chapter of this PeopleBook.

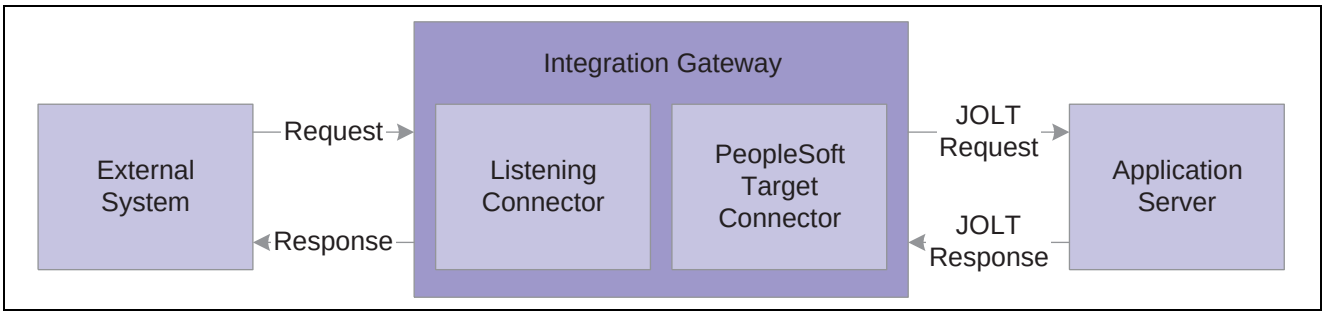
See Also

Chapter 3, “Understanding Messaging,” page 19

Incoming Request Flow

This section describes the flow of a typical request through PeopleSoft Integration Broker.

 Copyright © 1988-2007, Oracle. All rights reserved.



Incoming request through PeopleSoft Integration Broker

After the incoming request has been received by the integration gateway, the flow through PeopleSoft Integration Broker is the same, regardless of the listening connector used. With this in mind, no specific listening connector will be discussed here. The scenario is simple: a request is sent into the gateway, which then passes it on to the application server. The application server processes the request, and returns a response.

Step 1: External System Sends a Request to PeopleSoft Integration Broker

The first step is that an external system sends a request to PeopleSoft Integration Broker. The external system can be another PeopleSoft system or a third-party system.

Step 2: Request is Received by the Listening Connector

When a request is received by a listening connector, the first thing that the connector does is write the request to the gateway log file. (The gateway's integration properties file is used to set the logging level, which controls what is actually written to the log. If messages are not being seen in the log file, check to ensure that the log level is set correctly.) The request is written exactly as it is received. This is very useful in that it presents exactly what was sent on the wire, before the connector normalizes the service operation for use by the application server.

The connector then attempts to populate an internal request class with the particulars from the received request.

A term often used in conjunction with listening connectors is *credentials*. Incoming requests are thought to have two logical parts: the credentials and the body. The credentials can be thought of as the information required by PeopleSoft Integration Broker to process and deliver the payload of the message. The payload is located in the body. Since the credentials are separate from the body, the integration gateway does not need to parse or otherwise examine the request body for information on how to route it.

A request without credentials cannot be processed. If the integration gateway receives such a request an error will occur and an error message will be returned to the requestor.

Step 3: Request is Processed by the PeopleSoft Target Connector

In order for a request to be sent from the gateway to the application server, it must pass through the PeopleSoft target connector. This connector has two major responsibilities: it serializes the request to a string, and sends that string via a JOLT connection to the application server.

All communication between the gateway and the application server is done via the use of Multipurpose Internet Mail Extensions (MIME) messages. When the request is received by the connector, it builds a MIME message. Typically the MIME message will only have two sections. In the first, the credentials are stored in an XML document in a specific format. The second section stores the body.

At this point the request is in a standard format understood by both the gateway and the application server. All requests must eventually resolve to this format before they can be sent to the application server for processing. This format effectively isolates the application server from the protocols supported by the gateway; for the most part, there is no information present about what listening connector was initially invoked by the external request.

Copyright © 1988-2007, Oracle. All rights reserved.

One credential element that may be present is the one for cookies. Obviously if this is set, the application server would be right in assuming that the request came through the HTTP listening connector. Similarly, SOAP requests are passed into the application server in SOAP format. However, as a general rule the application server is isolated from the details of the protocol and the general broker code on the server does not care what listening connector was used for a given request.

Once the MIME message has been built, it is written to the gateway log.

Finally, the connector looks up the JOLT connection properties from the integration properties file and attempts to send the MIME to the application server. If these properties are not set up correctly, the gateway will be unable to send requests. This is a common source of error, so care should be taken when configuring this file.

An important point to keep in mind is that even though the MIME request to the application server may appear in the gateway log file, the actual request may not have made it to the application server, since the log entry is written before the service operation is sent. If a communication error occurs, the entry will still be present in the log file. However, if this situation occurs an exception will be thrown and an error log entry will also be created.

Step 4: Request is Received by the Application Server

When the MIME request is received by the application server, the system parses it into a request object. The MIME structure is not propagated into the server.

Assuming the request parses without error, the application server pre-processes it.

Pre-processing involves:

- Authenticating the service operation , depending on the authentication scheme configured. If the request fails authentication, an error is returned.
- Determining the direction of the service operation, by looking at the external alias on the routing definition that is associated with the service operation.
- Determining the runtime handler to invoke. Currently, there are three handler types supported by the integration broker: Ping, Synchronous, and Asynchronous. The service operation type determines the handler code to invoke. Synchronous service operations are passed to sync-specific code, and asynchronous service operations are passed to the publish/subscribe subsystem.

Once a request has been passed to its respective handler, further processing is dictated by the data and PeopleCode specific to a particular system. Or in the case of hub configurations, the request may immediately be routed to another external system.

Step 5: Response is Returned by the Application Server

Regardless of how the request is processed, a response must be returned by the application server to the gateway in the same thread of execution. The connection between the gateway and the application server is essentially synchronous, independent of the type of the service operation type. When the gateway sends a request to the application server, it expects and must get a response.

In the case of synchronous processing, the generation of the response is blocked by the processing of the request. A response cannot be generated until the service operation runs to completion. There may be a noticeable delay in receiving the response, depending on the processing required by the OnRequest method or if the request is being sent out of the broker to an external system for additional processing.

Asynchronous requests behave differently. Unlike synchronous requests, there is no blocking. A response is generated for an asynchronous request as soon as the request is placed on the publication queue. Because of this, a response generated for an asynchronous request is not a response in the strictest sense of the term. Such responses should really be considered acknowledgments that the pub/sub system has received the request. Receipt of such a response is not a guarantee that any applicable subscription PeopleCode has been successfully run.

 Copyright © 1988-2007, Oracle. All rights reserved.

Responses are converted to the MIME standard by the application server, and are returned to the gateway.

Step 6: Response is Received by the PeopleSoft Target Connector

As soon as the MIME response is received by the PeopleSoft target connector, it is written to the gateway log file.

The MIME response is then parsed back into a gateway request object, and is then returned to the listening connector.

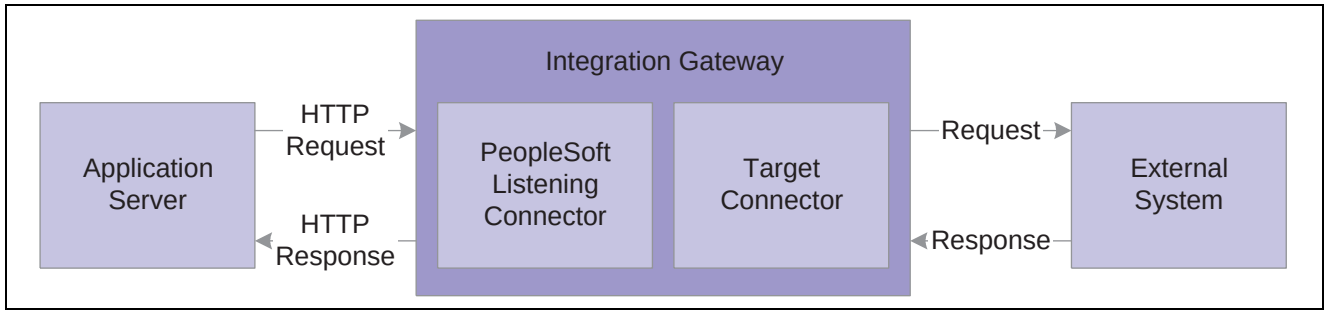
Step 7: Response is Received by the Listening Connector

The response object is returned to the listening connector, upon which the response is mapped to a response suitable for the given protocol.

It should be emphasized that, from the viewpoint of the listening connector, the processing of requests is done synchronously. A request is received by a listening connector which then coverts it to a suitable format, makes a blocking call to the gateway to handle the message, and ultimately gets a response back all in the same thread of execution.

Outgoing Request Flow

The following diagram shows an outgoing request through PeopleSoft Integration Broker.



Outgoing request through PeopleSoft Integration Broker

There are several scenarios that might result in a request being sent out of the broker. Requests can be sent in PeopleCode by using the Publish or SyncRequest methods of the Integration Broker class.

Regardless of how the request is created, the mechanism for sending it out of the broker is the same, and the flow is the same regardless of the specific outgoing target connector you invoke.

Step 1: Application Server Generates Request

Once an outgoing request has been generated, the application server must perform some basic processing before it can be sent out.

The application server looks at the request, and extracts the information about the node that it is being sent to.

If target connector information was not supplied via PeopleCode or as part of the routing, then the node name is then used to look up the name of the gateway to use, the target connector to use on that gateway, as well as any specific connector properties that need to be passed to the connector in order to handle the request. If this information is not found, an error will occur.

The application server modifies the outgoing request with the appropriate connector information.

The request is then converted to the MIME standard format, and is sent to the gateway over an HTTP connection.

Copyright © 1988-2007, Oracle. All rights reserved.

The request must be sent to the PeopleSoft listening connector on the gateway. The application server uses the value of the Gateway URL defined for the given gateway. If this URL is not valid or does not point to the PeopleSoft listening connector, the application server will be unable to send the request.

Step 2: Request is Received by the PeopleSoft Listening Connector

When the MIME request is received by the PeopleSoft listening connector, it is written to the gateway log file.

The request is converted from MIME format to a gateway request object.

The connector then examines the request to determine what target connector the request is to be sent to; that target connector is then invoked.

Step 3: Request is Received by the Target Connector

The target connector validates the request. Each connector requires certain properties to be set, otherwise the request cannot be sent. For example, the HTTP target connector requires that the PrimaryURL be set. If any mandatory connector properties are missing or are invalid, an error will be thrown.

The target connector then converts the request into whatever format is required by the protocol.

The modified request is then written to the gateway log, and then sent out.

Step 4: Response is Received by the Target Connector

The response received by the target connector is written to the gateway log, and the response is used to build a gateway response object, which is then returned to the PeopleSoft listening connector.

Step 5: Response is Received by the PeopleSoft Listening Connector

The response object is then converted to the MIME standard format by the connector.

The MIME response is then written to the gateway log file, and is then returned to the application server.

Interactions with the gateway are always synchronous. If a request is sent to the gateway, a response should be expected.

Step 2 is an HTTP POST request made of the gateway, and the response created here in Step 5 is returned in response to that HTTP request. The HTTP connection is open for the duration of the processing for that request.

The response object is returned to the listening connector, upon which the response is mapped to a response suitable for the given protocol.

 Copyright © 1988-2007, Oracle. All rights reserved.

CHAPTER 3

Understanding Messaging

This chapter discusses:

- Asynchronous messaging.
- Synchronous messaging.

Note. For compatibility with previous PeopleTools releases, the PeopleSoft Integration Broker services-oriented architecture introduced in PeopleTool 8.48 overlays the messaging architecture from earlier PeopleTools 8.4x releases.

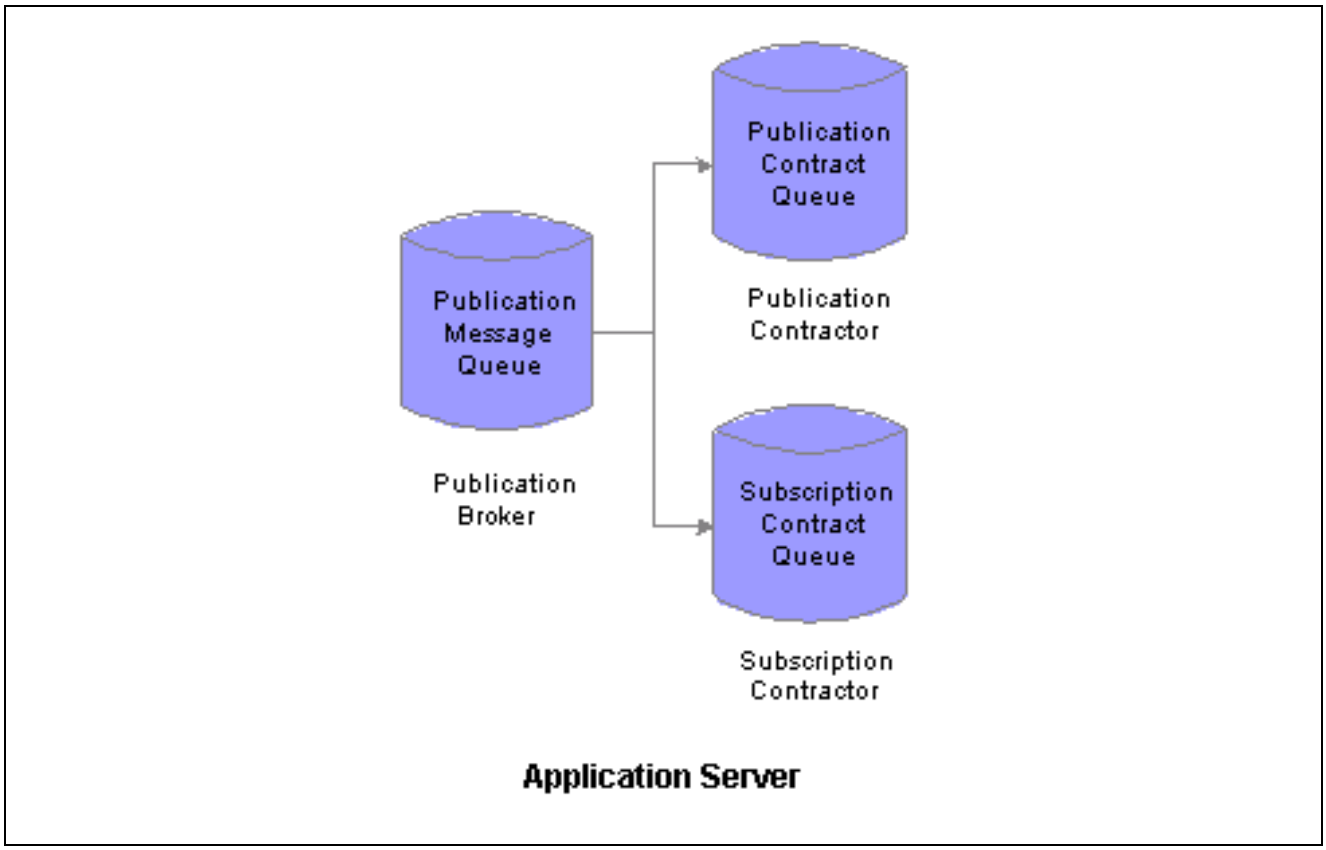
Asynchronous Messaging

This section discusses the PeopleSoft Integration Broker asynchronous messaging architecture.

Brokers, Contractors and Queues

The Publication Broker, Publication Contractor, and Subscription Contractor services are the primary application server elements required for asynchronous messaging. The Publication Broker service routes the workload to both contractor server processes, as illustrated in the following diagram:

Copyright © 1988-2007, Oracle. All rights reserved.



Brokers, contractors, and queues

Publication Broker Acts as the routing mechanism. When an asynchronous service operation arrives in its queue, the Publication Broker service runs the defined routing rules. If the service operation needs to be published to a remote node, it routes the service operation to the Publication Contractor service. If the service operation is subscribed to on the local node, then the Publication Broker routes the service operation to the Subscription Contractor service. Routing involves submitting either a subscription or publication contract to the appropriate contractor, followed by an asynchronous call to the contractor service notifying it that work is waiting in the queue.

Publication Contractor References the publication contract submitted by the Publication Broker service and performs an HTTP post of the publication service operation to the integration gateway. When the integration gateway sends a reply indicating that it received the publication service operation, the Publication Contractor service updates the publication contract with the status of subscription processing (*Done* or *Retry*).

Subscription Contractor References the subscription contract submitted by the Publication Broker service and runs the appropriate notification PeopleCode. Then it updates the subscription contract concerning the status of the subscription processing.

Messaging System Server Processes

The application server offers six server processes to handle asynchronous service operations. They work in pairs to provide three primary services:

 Copyright © 1988-2007, Oracle. All rights reserved.

Service	Server Processes
Publication Broker	• Broker Dispatcher (PSBRKDSP) • Broker Handler (PSBRKHND)
Publication Contractor	• Publication Dispatcher (PSPUBDSP) • Publication Handler (PSPUBHND)
Subscription Contractor	• Subscription Dispatcher (PSSUBDSP) • Subscription Handler (PSSUBHND)

Dispatchers and Handlers

Each of the Publication Broker, Publication Contractor, and Subscription Contractor is comprised of two individual server processes that work together to handle incoming requests. One server process functions as a dispatcher, while the other functions as a handler.

This relationship is analogous to the way that the application server handles workstation connections and requests. To handle the incoming client requests, the application server has a listener and a handler (or a pool of handlers). The listener receives the incoming requests and then routes them to an available handler.

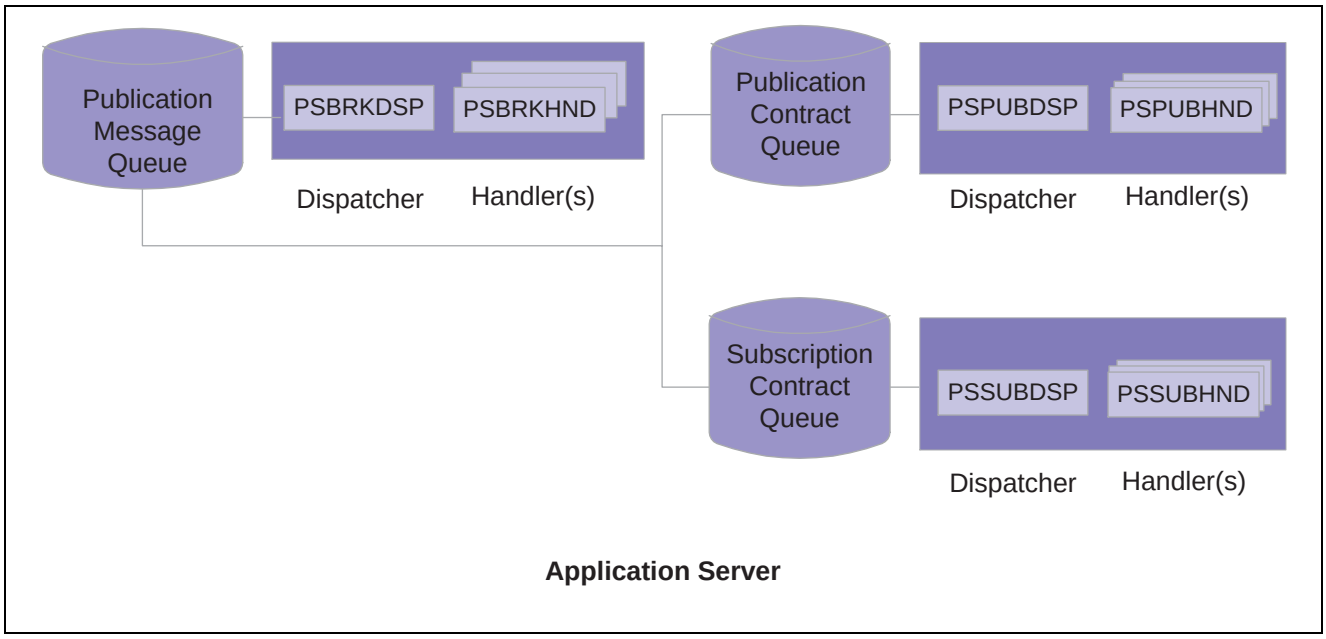
Typically, one listener serves many handlers. The relationship between the dispatcher and the handlers is analogous to the relationship between the Jolt Server Listener (JSL) and the Jolt Server Handler (JSH). In the case of the application messaging server processes, the dispatcher functions as the listener, and the handler as similar to the JSH.

For the services discussed in this section (Publication Contractor, Subscription Contractor, and Publication Broker) there are *at least* two server processes: a single dispatcher and one or more handlers. The PS*xxx*DSP server process is the dispatcher, and the PS*xxx*HND server process is the handler.

Note. The *xxx* represents BRK, PUB, or SUB. For example, in the case of the Publication Broker, PSBRKDSP is the dispatcher and PSBRKHND is the handler.

This diagram shows the messaging server processes grouped by their functions in the messaging architecture:

Copyright © 1988-2007, Oracle. All rights reserved.



Dispatchers and handlers

Asynchronous Service Operation Publication

This section discusses:

- Asynchronous publish of a service operation instance.
- Asynchronous publish of a publication contract.

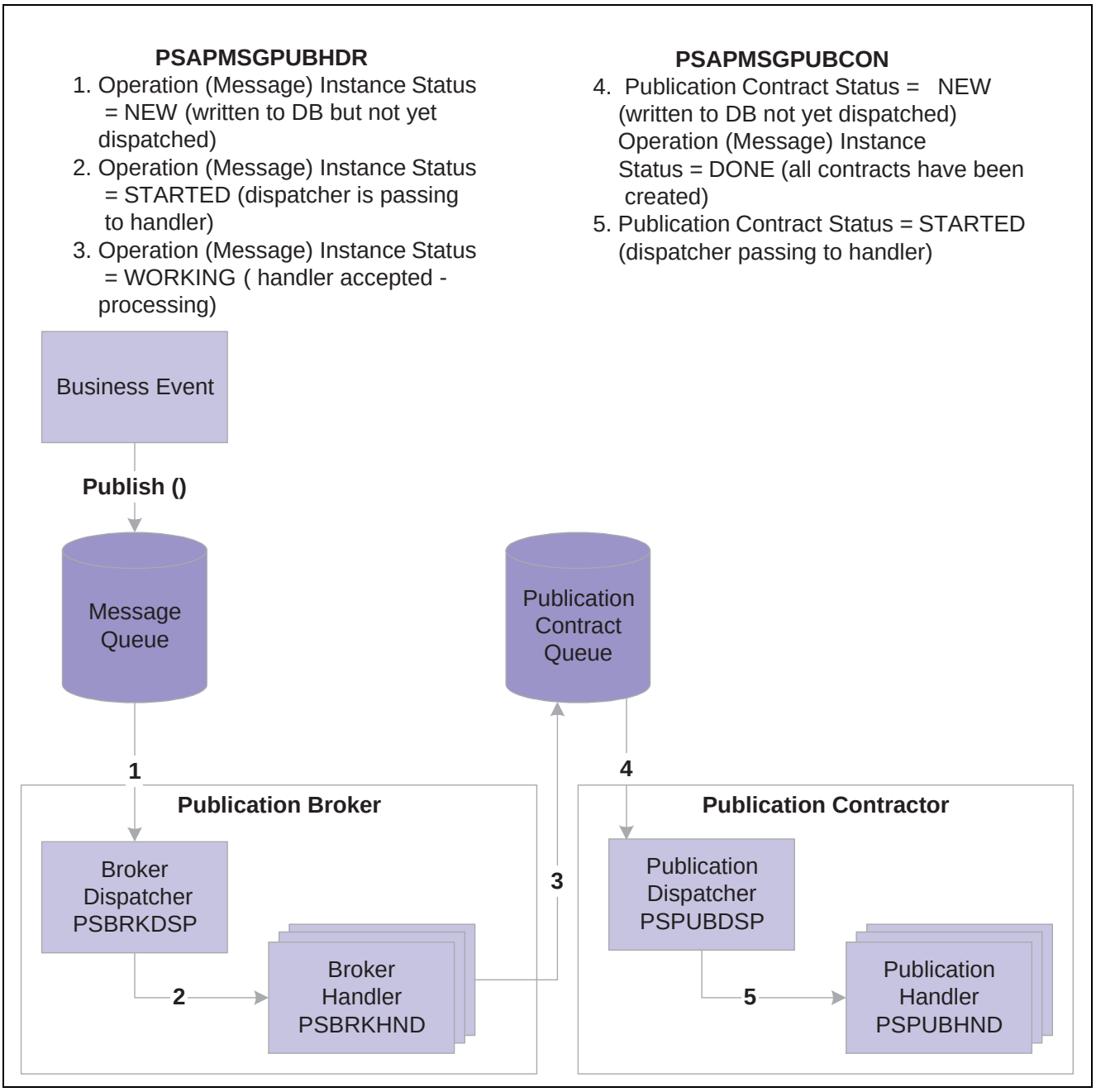
Understanding Asynchronous Service Operation Publication

This section describes the flow of an asynchronous service operation publication through PeopleSoft Integration Broker, as well as the status of the service operations as they appear in Service Operations Monitor.

Asynchronous Publish of Service Operation Instances

This diagram shows an asynchronous publish of a service operation instance:

Copyright © 1988-2007, Oracle. All rights reserved.



Asynchronous publication of an operation instance

The processing steps of an asynchronous publication of a service operation instance are:

1. The service operation is published and enters the message queue.

   The Broker Dispatcher process picks up the service operation instance from its queue. During this stage, the status of the service operation instance is *New*.

2. The Broker Dispatcher process passes the service operation instance to the Broker Handler process.

   During this stage, the status of the service operation instance is *Started*.

3. The Broker Handler process accepts the service operation instance, reads the data, and runs the routing rules to determine where the publication needs to be delivered.

Copyright © 1988-2007, Oracle. All rights reserved.

The Broker Handler process then writes a publication contract in the PSAPMSGPUBCON table and notifies the Publication Contractor service that it has an item to process. During this stage, the status of the service operation instance is *Working*.

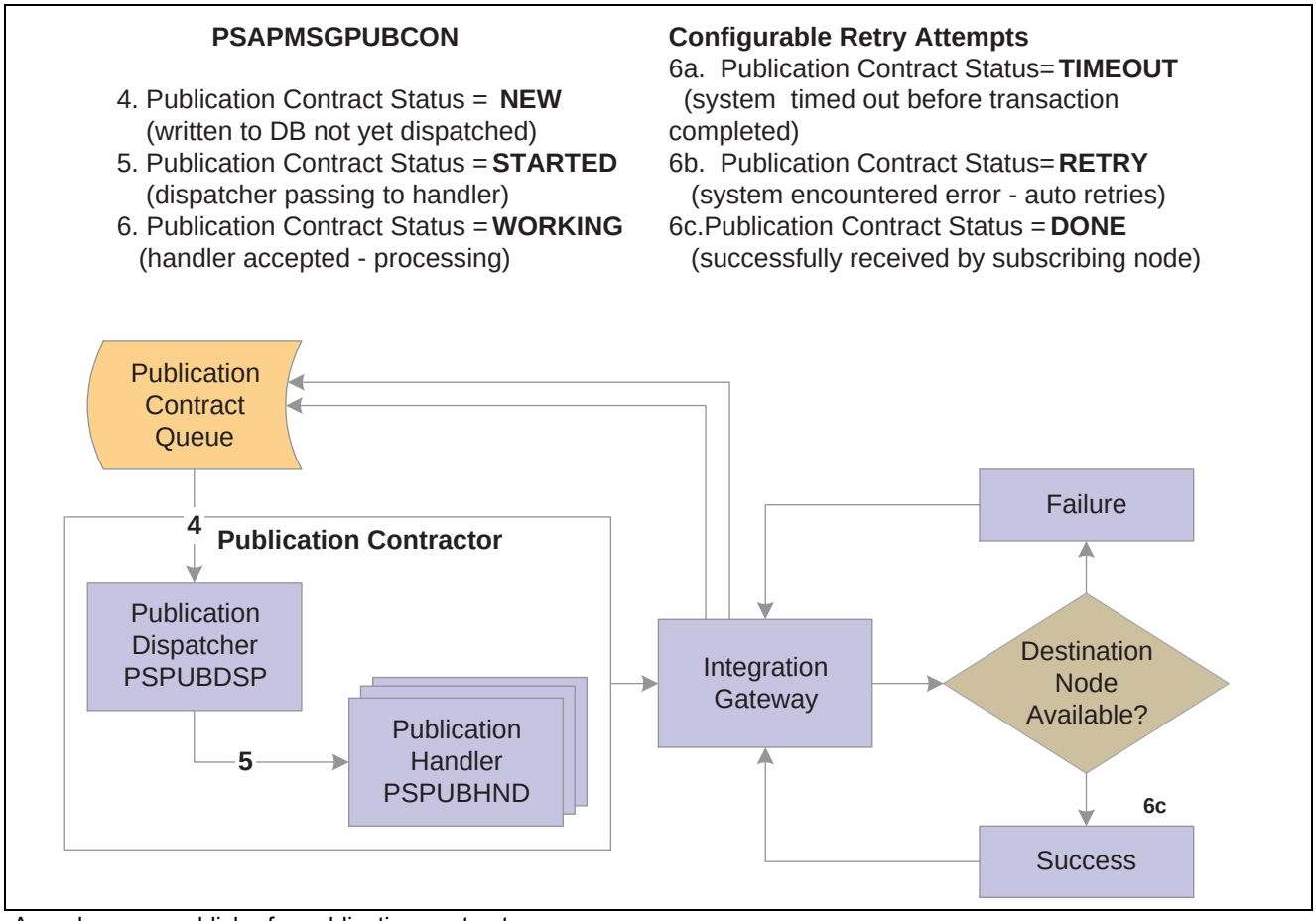
4. After the service operation is stored in the publication contact queue, the status of the publication contract is *New*, the service operation instance status is *Done*, and the Publication Dispatcher process picks up the publication contract from its queue.

You view service operation instance status on the Operation Instances page of the Service Operations Monitor. To access the page selectPeopleTools, Integration Broker, Service Operations Monitor, Monitor, Asynchronous Services, Operation Instances.

See Chapter 21, "Using the Service Operations Monitor," Monitoring Asynchronous Service Operation Instances, page 423.

Asynchronous Publish of Publication Contracts

This diagram shows the flow of an asynchronous publication contract:



Asynchronous publish of a publication contract

The processing steps of an asynchronous publish of a publication contract are:

1. The Publication Dispatcher process passes the publication contract to the Publication Handler process.

   At this stage the status of the publication contract is *Started*.

2. The Publication Handler process accepts the publication contract and attempts to deliver the service operation to the integration gateway.

 Copyright © 1988-2007, Oracle. All rights reserved.

At this stage, the status of the publication contract is *Working*.

If the publication contract is successfully delivered to the destination node, the status is *Done* (refer to step 6c in the diagram).

If an error occurs during this stage, the status is *Error*.

If the system times out before the transaction is completed, the status is *Timeout* (6a in the diagram).

If the delivery fails, the Publication Handler process retries the delivery, and the status is *Retry* (refer to step 6b in the diagram). When service operations have *Retry* status, the service operations are not resent until an internal ping is successful. This ping is similar to a node ping. The Publication Contract Dispatcher, as part of its on idle processing, pings a node that is in *Retry* status and verifies if the connection is reestablished. When the ping is successful the Publication Contract Dispatcher resends the service operation. The service operation goes back to the Publication Handler process and returns to *Working* status.

You can view the status information for the publication contract using the Publication Contracts page in the Service Operations Monitor. To access the page, select PeopleTools, Integration Broker, Service Operations Monitor, Monitor, Asynchronous Services, Publication Contracts.

See Chapter 21, "Using the Service Operations Monitor," Monitoring Publication Contracts, page 424.

Asynchronous Service Operation Subscription

This section discusses:

- Asynchronous subscription of a service operation instance.
- Asynchronous subscription contracts.

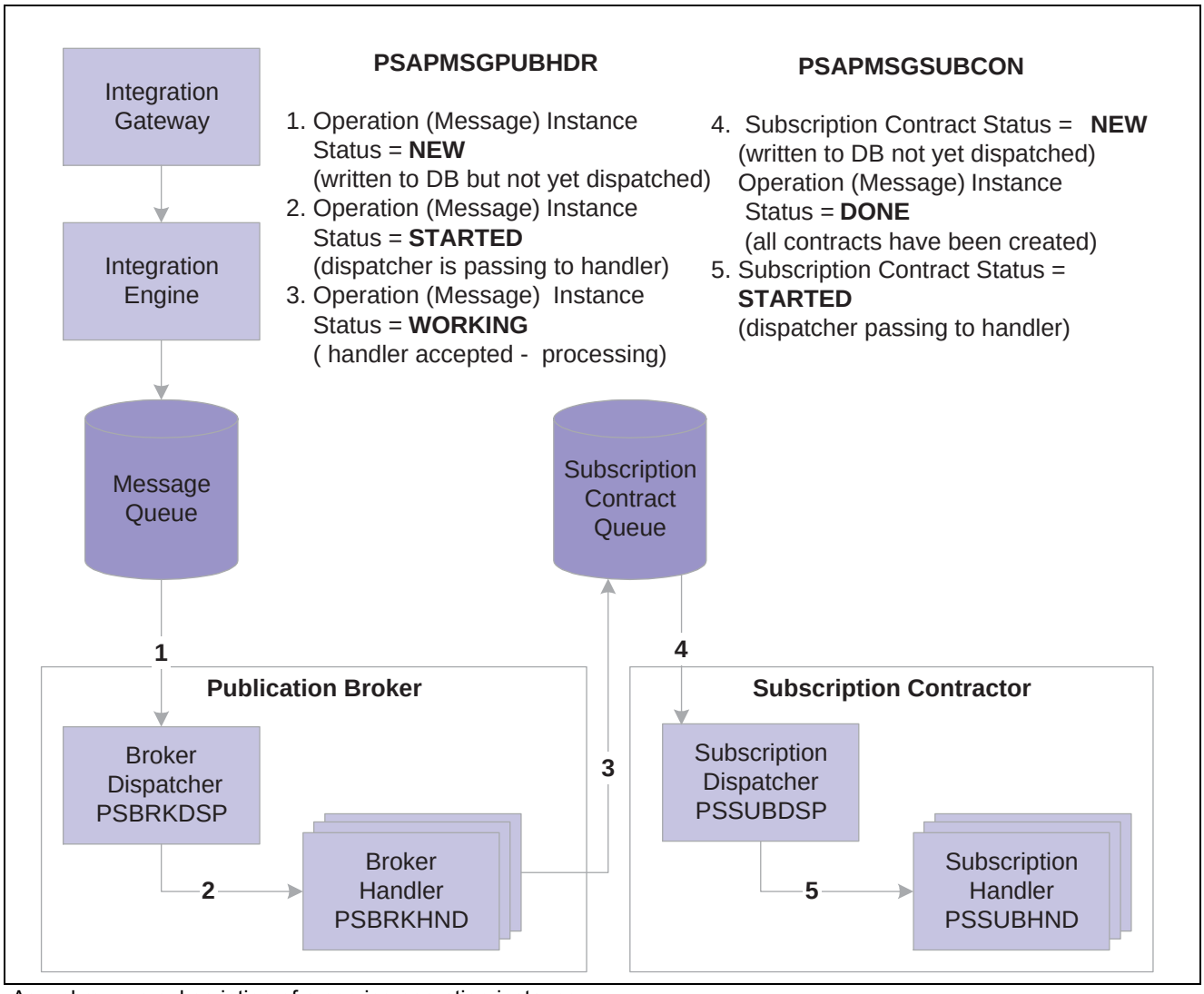
Understanding Asynchronous Service Operation Subscription

This section describes the flow of an asynchronous service operation subscription through PeopleSoft Integration Broker, as well as the service operation status at each stage of the process.

Asynchronous Subscription of Service Operation Instances

This diagram illustrates the flow of an asynchronous service operation subscription through PeopleSoft Integration Broker:

Copyright © 1988-2007, Oracle. All rights reserved.



Asynchronous subscription of a service operation instance

The processing steps are:

1. The service operation enters the message queue.

   The Broker Dispatcher process picks up the service operation instance from its queue. During this stage, the status of the service operation instance is *New*.

2. The Broker Dispatcher process passes the service operation instance to the Broker Handler process.

   During this stage, the status of the service operation instance is *Started*.

3. The Broker Handler process accepts the service operation instance, reads the data, and runs the subscription routing rules to determine if the service operation needs to be processed locally.

   The Broker Handler process then writes a subscription contract in the PSAPMSGPUBCON table (the subscription contract queue) and notifies the Subscription Contractor service that it has an item to process. During this stage, the status of the service operation instance is *Working*.

Once the service operation is stored in the subscription contact queue, the status of the subscription contract is *New*, the service operation instance status is *Done,* and the Subscription Dispatcher process picks up the subscription contract from its queue. In this example, at the point when the status of the asynchronous service operation instance is *Done*, the subscription contract status is *New*.

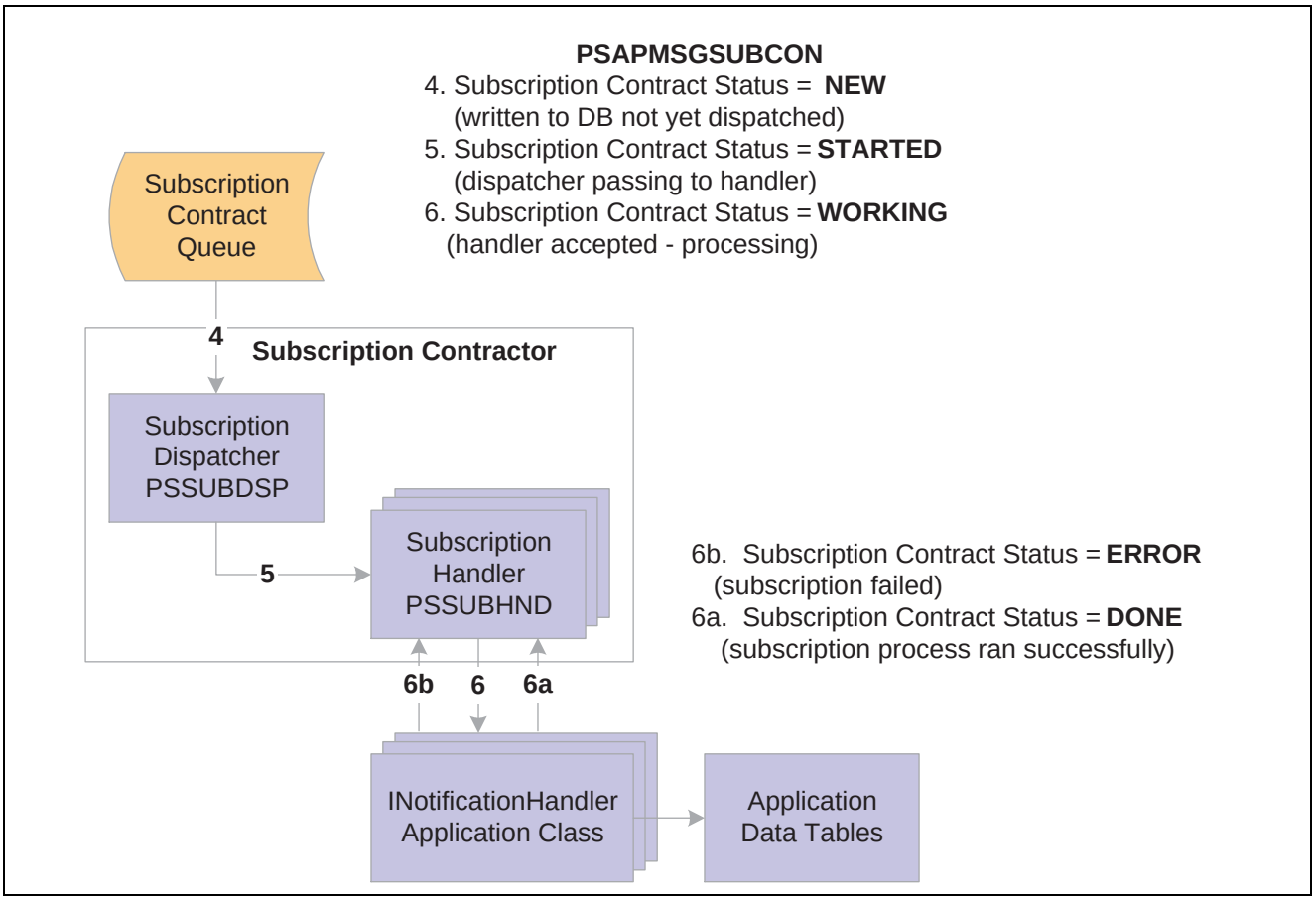
 Copyright © 1988-2007, Oracle. All rights reserved.

You view service operation instance status on the Operation Instances page of the Service Operations Monitor. To access this page, select PeopleTools, Integration Broker, Service Operations Monitor, Monitor, Asynchronous Services, Operation Instances.

See Chapter 21, "Using the Service Operations Monitor," Monitoring Asynchronous Service Operation Instances, page 423.

Asynchronous Subscription Contract

This diagram shows the flow of an asynchronous subscription contract:



Asynchronous subscription contract

The processing steps are:

1. The Subscription Dispatcher process passes the subscription contract to the Subscription Handler process.

   At this stage the status of the subscription contract is *Started*.

2. The Subscription Handler process accepts the subscription contract and runs the notification PeopleCode.

   In the example shown in the diagram, the notification PeopleCode then uses the service operation data to update application data tables. However, the notification PeopleCode can use the service operation data as input to look up information, create and publish another service operation, and so forth. At this stage, the status of the publication contract is *Working*.

   If the notification PeopleCode runs successfully, the status is *Done* (refer to step 6a in the diagram). If an error occurs during this stage, the status is *Error* (refer to step 6b in the diagram).

Copyright © 1988-2007, Oracle. All rights reserved.

To view status information for subscription contracts, use the Subscription Contracts page in the Services Operation Monitor. To access the page select PeopleTools, Integration Broker, Service Operations Monitor, Monitor, Asynchronous Services, Subscription Contracts.

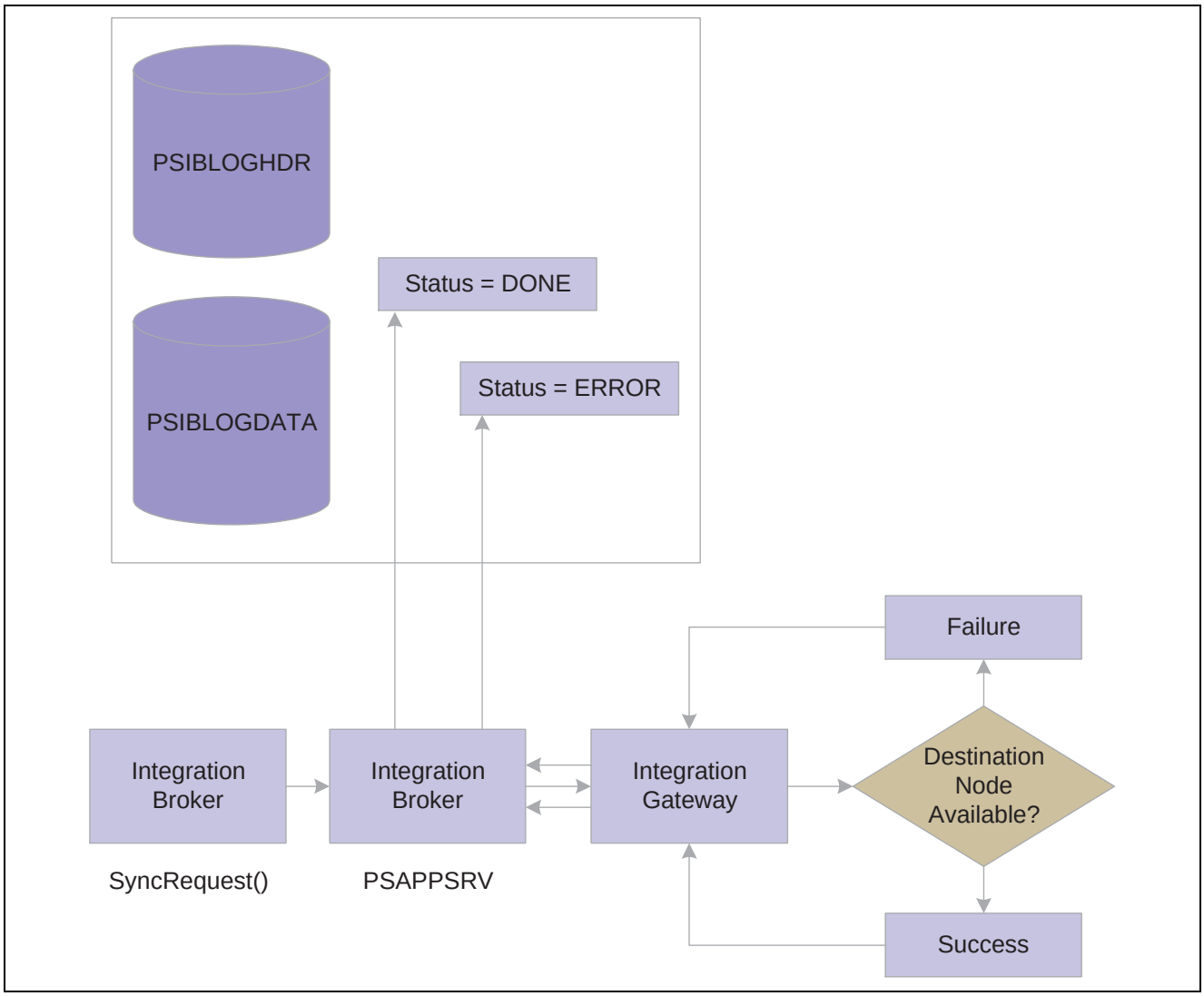
See Chapter 21, "Using the Service Operations Monitor," Monitoring Subscription Contracts, page 425.

Synchronous Messaging

This section discusses synchronous messaging in PeopleSoft Integration Broker.

Synchronous Service Operation Publication

This diagram illustrates using PeopleSoft Integration Broker to consume a synchronous service operation:



Synchronous service operation publication

The processing steps are:

Copyright © 1988-2007, Oracle. All rights reserved.

1. The system makes a SyncRequest ( ) call to PeopleSoft Integration Broker.
2. PeopleSoft Integration Broker sends the service operation to the integration gateway.
3. If the integration gateway can deliver the service operation to the destination node, the process is successful and the status is *Done*.

   If the process in unsuccessful, the status is *Error*.

   For PeopleSoft Integration Broker to shoe the status, logging on the routing definition for the service operation must be set.

You can view the status information for the invocation in the Service Operations Monitor using the Synchronous Services page. To access the page select PeopleTools, Integration Broker, Service Operations Monitor, Monitor, Synchronous Services.

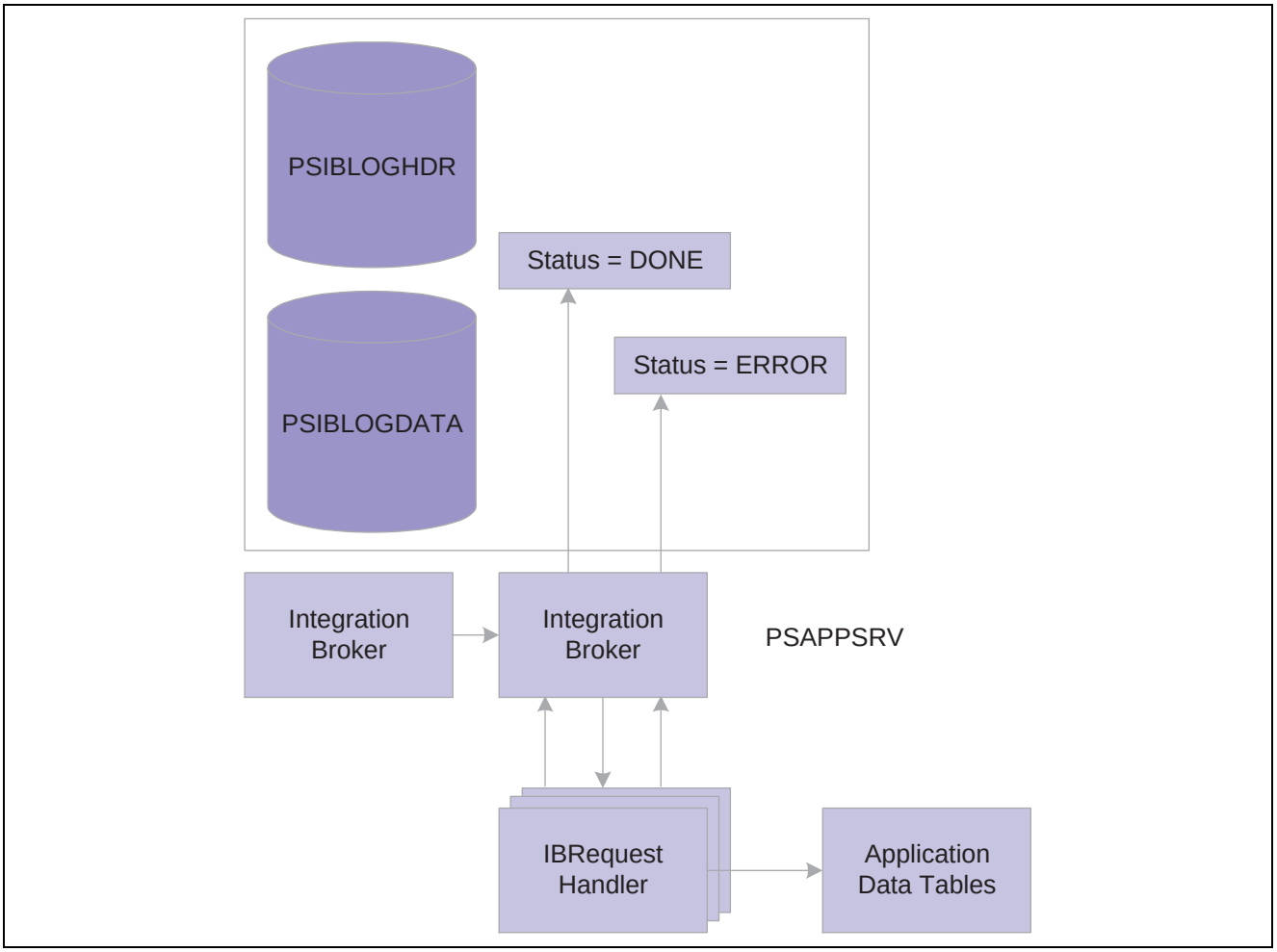
See Also

Chapter 21, “Using the Service Operations Monitor,” Monitoring Synchronous Service Operations, page 435

Chapter 18, “Managing Routing Definitions,” Defining General Routing Information, page 342

Synchronous Service Operation Subscription

This diagram illustrates providing a synchronous service operation through PeopleSoft Integration Broker:



Synchronous service operation subscription

Copyright © 1988-2007, Oracle. All rights reserved.

1. The integration gateway passes an inbound synchronous service operation to the integration engine.
2. The integration engine runs an OnRequest PeopleCode program.
3. The OnRequest PeopleCode program attempts to update the application data tables.

   If the program runs successfully, the status is *Done*. If the OnRequest PeopleCode program fails, the status is *Error*.

   For PeopleSoft Integration Broker to shoe the status, logging on the routing definition for the service operation must be set.

You can view the status information for the publication in the Service Operations Monitor by using the Synchronous Services page. Access this page by selecting PeopleTools, Integration Broker, Service Operations Monitor, Monitor, Synchronous Services.

See Also

Chapter 21, “Using the Service Operations Monitor,” Monitoring Synchronous Service Operations, page 435

Chapter 18, “Managing Routing Definitions,” Defining General Routing Information, page 342

 Copyright © 1988-2007, Oracle. All rights reserved.

CHAPTER 4

Understanding Creating and Implementing Integrations

This chapter provides a high-level overview of the steps to set up and create integrations and discusses:

- Determining the messaging architecture.
- Installing web servers.
- Installing PeopleTools.
- Installing application databases.
- Installing and starting the PeopleSoft Pure Internet Architecture and integration gateways
- Installing application servers.
- Starting the PeopleSoft Pure Internet Architecture.
- Configuring and starting messaging servers for asynchronous messaging.
- Activating pub/sub server domains.
- Defining integration gateways and loading connectors.
- Configuring integration gateway properties.
- Configuring PeopleSoft Integration Broker to handle services.
- Creating integration metadata.
- Granting security access to service operations

See Also

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

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 PeopleSoft 8.4x systems where a firewall is involved, integration with third-party systems, or integrations with PeopleSoft 8.1x systems.

This PeopleBook features an appendix that provides overview information about messaging architecture scenarios. It discusses the architecture for integrating with:

- PeopleSoft Integration Broker systems.
- PeopleSoft Integration Broker systems through firewalls.

Copyright © 1988-2007, Oracle. All rights reserved.

- PeopleSoft Integration Broker systems by using hubs.
- Third-party systems.
- Third-party systems by using remote gateways.
- PeopleSoft 8.1x systems.

See Also

Appendix A, “Integration Scenarios,” page 679

Installing Web Servers

To install and run PeopleTools, you must install a web server. PeopleTools currently supports BEA WebLogic 8.1, IBM WebSphere 5.1 and Oracle Application Server 10g.

See Also

Your web server documentation

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 and the integration gateway.

See Also

PeopleTools 8.49 Install Guide for your database.

Installing Application Databases

After you install PeopleTools, install your application database.

See Also

PeopleTools 8.49 Install Guide for your database.

Starting the PeopleSoft Pure Internet Architecture

The next step is to install and start the PeopleSoft Pure Internet Architecture. The integration gateway is installed as part of the PeopleSoft Pure Internet Architecture installation.

1. Run the PeopleSoft Pure Internet Architecture setup program and start the application.

 Copyright © 1988-2007, Oracle. All rights reserved.

The location of the PeopleSoft Pure Internet Architecture setup file is <PS_HOME>\setup \PsMpPIAInstall\setup.exe.

2. Verify that the web server is running; it must be running to start the PeopleSoft Pure Internet Architecture.
3. Start the PeopleSoft Pure Internet Architecture by launching the startPIA.cmd file.

   The location of the file is <PS_HOME>webserv\peoplesoft\startPIA.cmd.

   Note. To stop the PeopleSoft Pure Internet Architecture, launch the stopPIA.cmd file. The location of the file is <PS_HOME>webserv\peoplesoft\stopPIA.cmd.

4. Verify that the PeopleSoft Pure Internet Architecture is installed correctly by launching it in a web browser.

See Also

PeopleTools 8.49 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 Also

Chapter 6, “Administering Messaging Servers for Asynchronous Messaging,” page 41

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, access the Quick Configuration page. In the Integration Broker Domains section of the page, locate your machine name and select Active from the dropdown list and click the Save button.

See Also

Chapter 5, “Using the Integration Broker Quick Configuration Page,” page 37

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.

Copyright © 1988-2007, Oracle. All rights reserved.

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://<local_host>/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 5, "Using the Integration Broker Quick Configuration Page," page 37

Chapter 7, "Managing Integration Gateways," page 53

Configuring Integration Gateway Properties

After you define the default local integration gateway, specify the integration gateway URL and load the delivered connectors, you must configure the integration gateway properties file.

To establish settings for the integration gateway and its delivered connectors, you use the integrationGateway.properties file. To access the properties file, in the Gateways component click the Properties link next to the integration gateway URL field.

At a minimum you must set the BEA 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."

See Also

Chapter 7, "Managing Integration Gateways," Defining Integration Gateways, page 55

Configuring PeopleSoft Integration Broker 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

Chapter 14, "Managing Services," Configuring PeopleSoft Integration Broker for Handling Services, page 277

 Copyright © 1988-2007, Oracle. All rights reserved.

Creating Integration Metadata

This section provides a high-level overview of the integration metadata that you need to create to use PeopleSoft Integration Broker and a suggested order for creating it.

Understanding Integration Metadata

You use the following integration metadata to create and implement integrations.

Integration PeopleCode — You use integration PeopleCode to send and receive messages, route messages and manipulate message content.

Integration gateway definitions — This definition is an application's internal representation of an installed integration gateway. An application requires at least the local gateway, through which it can send and receive messages. Multiple nodes can share the same local gateway, which might be the only gateway that you need for all of the integrations.

Message definitions — Message definitions provide the physical description of the data that is being sent, including fields, field types, and field lengths.

Node definitions — 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

Queue definitions — A queue isolates different groups of service operations from each other.

Routing definitions — Routing definitions determine the sender and receiver of an integration. Routing definitions allow you to specify inbound and outbound transformations that enable you to transform data structures into those that the sending or receiving systems can understand.

Service definitions — Service definitions group service operations into logical groups or categories.

Service operation definitions — Service operations define the processing logic of an integration. They specify the inbound, outbound and fault messages associated with an integration, the integration PeopleCode to invoke, and the routing to use.

Transformation programs — A transformation or transform program is a type of Application Engine program that you develop and specify as part of a routing definition. PeopleSoft Integration Broker supports the use of Extensible Stylesheet Language Transformation (XSLT) code and PeopleCode for developing transform programs.

Transform programs can transform, filter and translate data.

Copyright © 1988-2007, Oracle. All rights reserved.

Order of Precedence for Creating Integration Metadata

Create integration metadata in the following order:

1. Integration gateway definition.
2. Node definition.
3. Message definition.
4. Integration PeopleCode.
5. Transformation programs.
6. Queue definition.
7. Service definition.
8. Service operation definition.
9. Routing definition.

Granting Security Access Service Operations

When you create service operations, you must grant permission list access to them .

Moreover, granting security access to service operations is required to access and modify integration information in the Service Operations Monitor.

To grant security access to service operations, after you create and save a service operation a Service Operation Security link displays on the Service Operations-General page. Click the link to access the Web Service Access page and assign a permission list to the service operation.

See Also

Chapter 15, “Managing Service Operations,” Setting Permissions to Service Operations, page 304

 Copyright © 1988-2007, Oracle. All rights reserved.

CHAPTER 5

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.

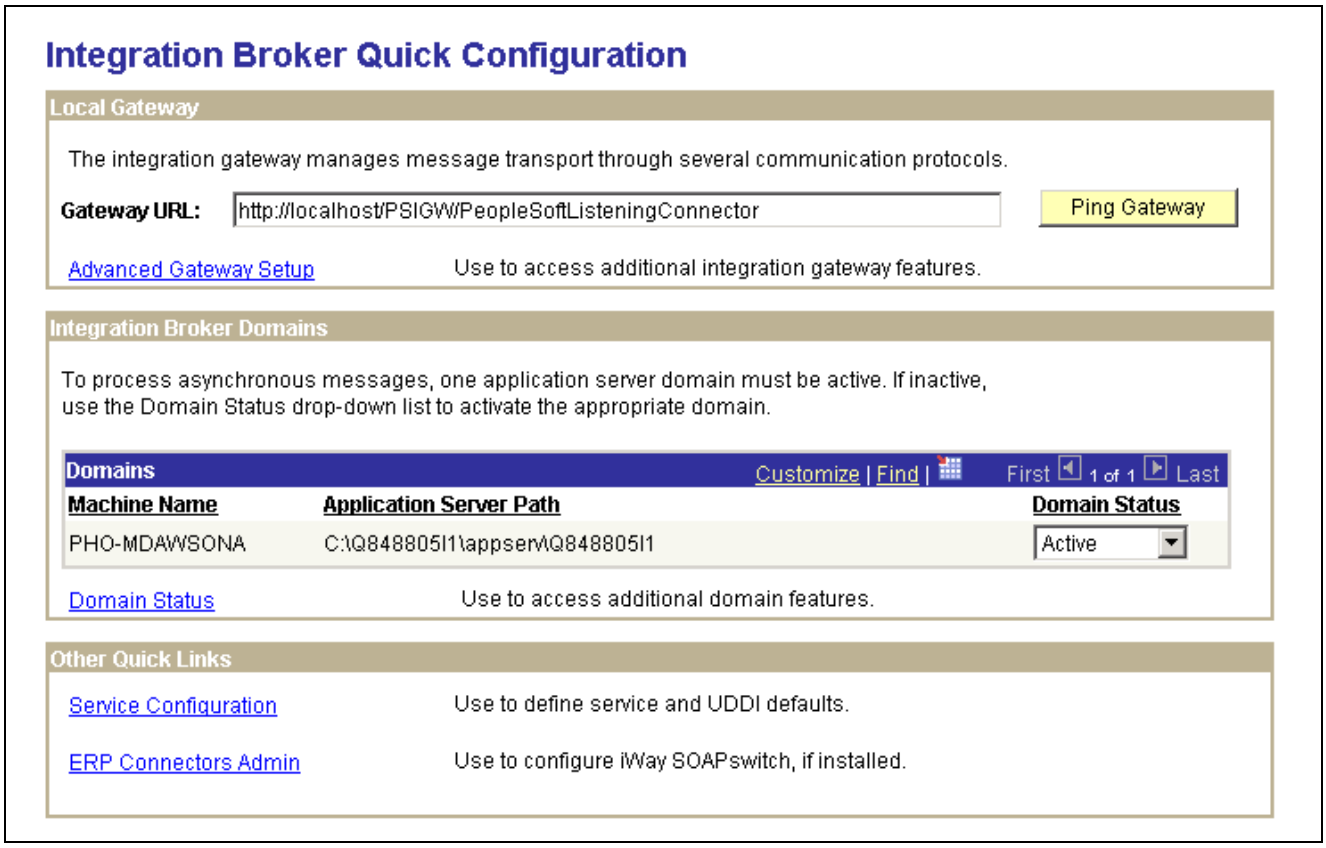
See Also

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.

Copyright © 1988-2007, Oracle. All rights reserved.



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 7, "Managing Integration Gateways," Setting General Connection Properties, page 69.

Domain Status Click Active from the dropdown list to activate pub/sub servers on application server domains.

Copyright © 1988-2007, Oracle. All rights reserved.

	You must activate the pub/sub servers on application server domains used for messaging before you can use them to successfully to send and receive messages. The dropdown list 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 21, "Using the Service Operations Monitor," Managing Pub/Sub Server Domains, page 477.
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 14, "Managing Services," Configuring PeopleSoft Integration Broker for Handling Services, page 277.
ERP Connectors Admin	Click the link to access the ERP Connectors Admin page to set required properties and login ID/password for using the iWay SOAPswitch connectors for integrating with third-party ERP systems. See Chapter 26, "Integrating with ERP Systems," page 571.

See Also

Chapter 4, "Understanding Creating and Implementing Integrations," page 31

Copyright © 1988-2007, Oracle. All rights reserved.

 Copyright © 1988-2007, Oracle. All rights reserved.

CHAPTER 6

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 BEA 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 active 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 Also

Chapter 5, “Using the Integration Broker Quick Configuration Page,” page 37

Chapter 21, “Using the Service Operations Monitor,” Managing Pub/Sub Server Domains, page 477

Copyright © 1988-2007, Oracle. All rights reserved.

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:

 Copyright © 1988-2007, Oracle. All rights reserved.

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 8.49 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.

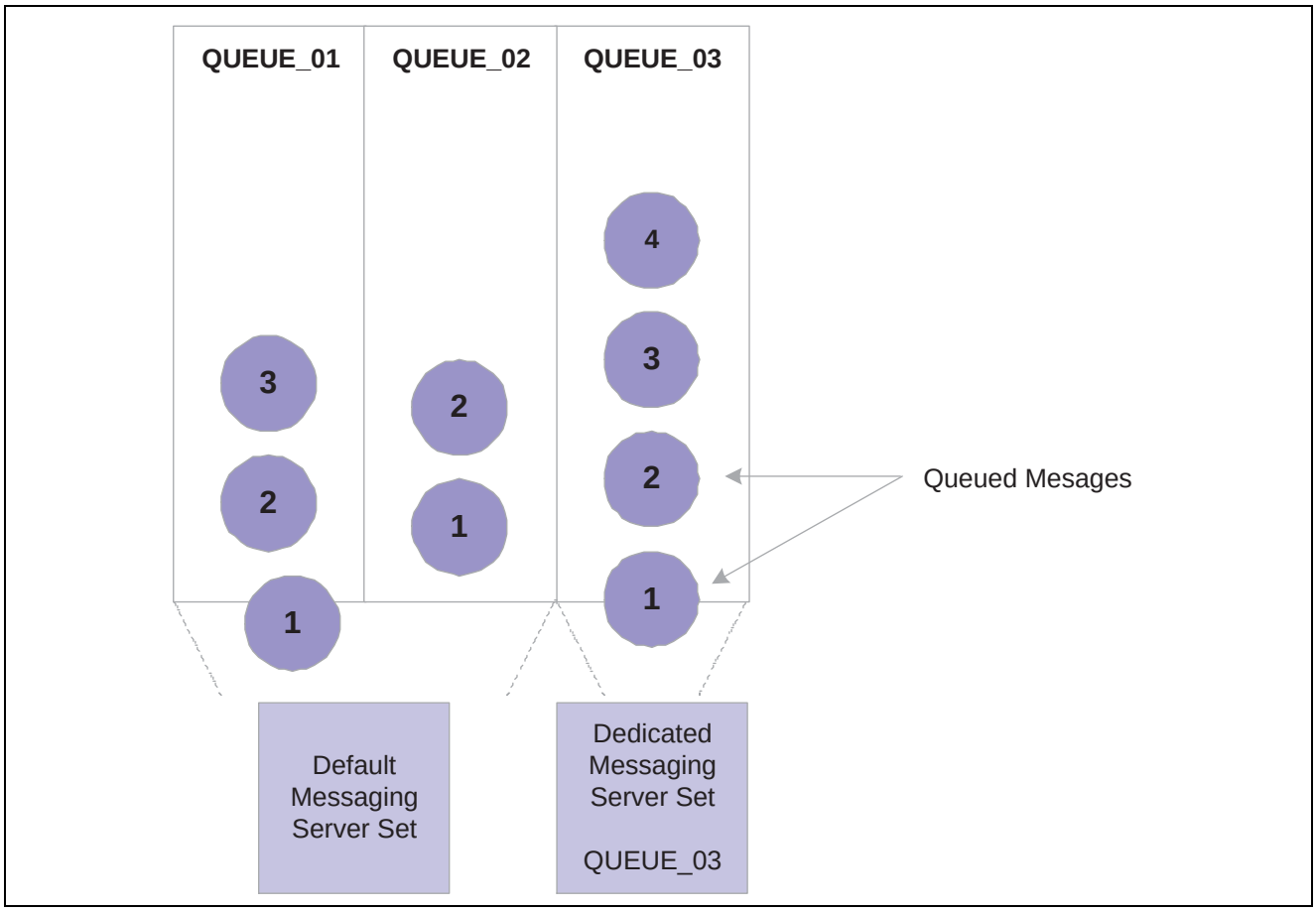
You might use the default messaging server set as the only messaging server set; however, in most cases, it is insufficient. As the volume of published messages increases in a production system, it's likely that a single messaging server set will become overloaded. To avoid potential overloads and performance 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:

Copyright © 1988-2007, Oracle. All rights reserved.



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.

Copyright © 1988-2007, Oracle. All rights reserved.

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 BEA tuxedo queue size is large enough and correctly configured in PSADMIN.

  See Chapter 6, "Administering Messaging Servers for Asynchronous Messaging," Setting the BEA Tuxedo Queue Size, page 52.
- 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 21, "Using the Service Operations Monitor," Setting Up Domain Failover, page 480.
- 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 6, "Administering Messaging Servers for Asynchronous Messaging," Specifying Dispatcher Parameters, page 48.

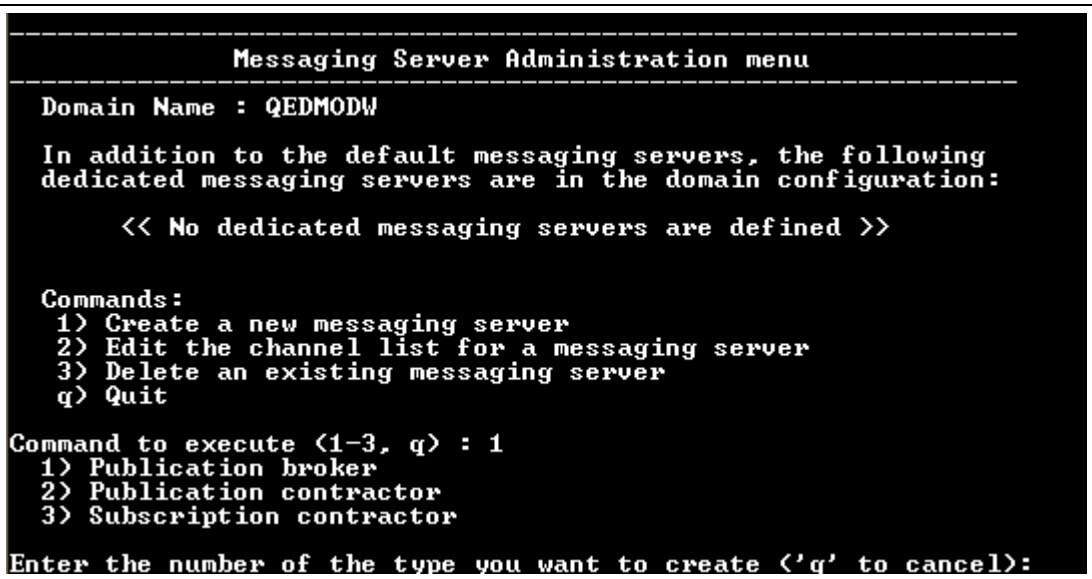
Creating and Assigning Dedicated Servers

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

Copyright © 1988-2007, Oracle. All rights reserved.

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:



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.

 Copyright © 1988-2007, Oracle. All rights reserved.

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.

Copyright © 1988-2007, Oracle. All rights reserved.

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 8.49 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.

 Copyright © 1988-2007, Oracle. All rights reserved.

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 BEA 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.

Copyright © 1988-2007, Oracle. All rights reserved.

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 BEA 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*

 Copyright © 1988-2007, Oracle. All rights reserved.

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 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 BEA 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 BEA 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.

Copyright © 1988-2007, Oracle. All rights reserved.

Setting the BEA 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 *Enterprise PeopleTools 8.49 PeopleBook: System and Server Administration*, “Using the PSADMIN Utility” and *Enterprise PeopleTools 8.49 PeopleBook: System and Server Administration*, “Using PSADMIN Menus”.

 Copyright © 1988-2007, Oracle. All rights reserved.

CHAPTER 7

Managing Integration Gateways

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

- Administer integration gateways.
- Use the integrationGateway.properties file.
- Encrypt passwords.
- Configure security and general properties.
- Configure integration gateways for load balancing.
- Apply message transformations at the integration gateway.
- Bypass integration engines to send messages.

Understanding Integration Gateway Configuration

This section discusses:

- Local gateway compatibility.
- Types of integration gateway configuration.

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 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.

Copyright © 1988-2007, Oracle. All rights reserved.

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 27, "Setting Up Secure Integration Environments," Installing Web Server-Based Digital Certificates, page 603.

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 7, "Managing Integration Gateways," Administering Integration Gateways, page 54.

Minimum Integration Gateway Setup Requirements

The minimum setup requirement to run an integration gateway are:

1. Specify the gateway URL.
2. Specify the BEA Jolt connection string properties to enable communication with each PeopleSoft Integration Broker node that will be involved in an integration that uses a gateway.

You can perform these tasks using the Integration Broker Quick Configuration page.

See Also

Chapter 5, "Using the Integration Broker Quick Configuration Page," page 37

Administering Integration Gateways

This section discusses how to:

- Define integration gateways.

 Copyright © 1988-2007, Oracle. All rights reserved.

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

Pages Used to Administer Integration Gateways

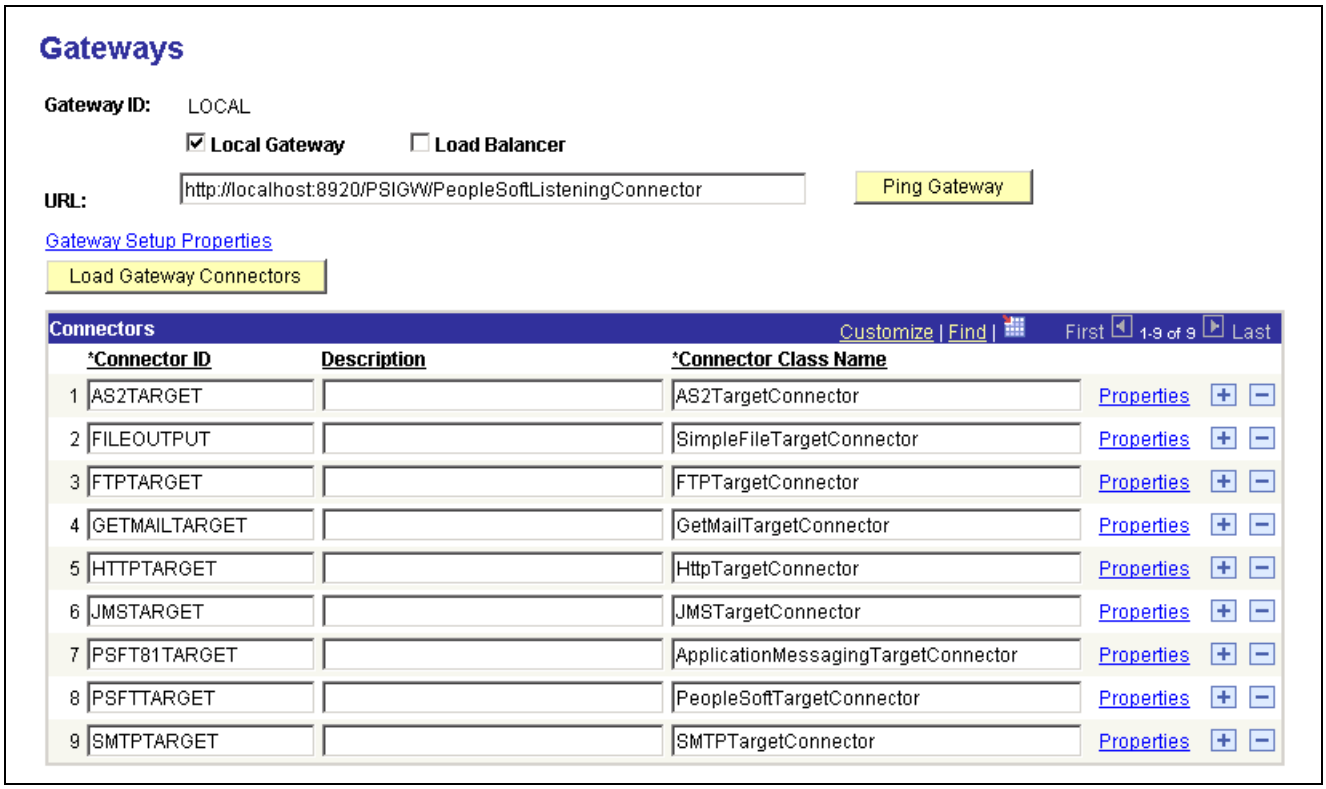
Page Name	Object Name	Navigation	Usage
Gateway page	IB_GATEWAY	Select PeopleTools, Integration Broker, Configuration, Gateways.	Use this page to: • Define an integration gateway. • Configure an integration gateway. • Ping an integration gateway. • Load target connectors.
Connector Properties page	IB_CONNPROP	From the Gateway page, in the Connectors grid locate a target connector with which to work. Click the Properties link for the connector.	Modify target connector properties.

Defining Integration Gateways

Use the Gateway page in the Gateways component to specify the location of the gateway, update configuration settings, and register target connectors to be used with the gateway. To access the page, select PeopleTools, Integration Broker, Configuration, Gateways.

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.

Copyright © 1988-2007, Oracle. All rights reserved.



Gateways page with the URL defined and target connectors loaded

To define and configure a gateway:

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

   The Gateways search page appears. Do one of the following:

   - 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

Copyright © 1988-2007, Oracle. All rights reserved.

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 28, "Tuning Messaging System Performance," Configuring Integration Gateways for Load Balancing, page 657

Pinging Integration Gateways

You can 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. 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.

Copyright © 1988-2007, Oracle. All rights reserved.

See Appendix D, "Using the Integration Broker Connector SDK," page 751.

Click the Load Gateway Connectors button on the Gateways page 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. Add a new row in the Connectors grid of the Gateways page.
2. Enter the ID for the new connector.
3. Enter the connector class name.
4. Click Properties to edit the connector's properties.

See Also

Appendix B, "Using the Delivered Listening Connectors and Target Connectors," page 705

Appendix D, "Using the Integration Broker Connector SDK," page 751

Refreshing Integration Gateway Properties

When the gateway server boots, it reads and applies the properties in the integrationGateway.properties configuration file. Changes that you make to the file while the server is running have no immediate effect.

If you make changes to the integrationGateway.properties file while the server is running, you can click the OK button on the Gateways Properties page. PeopleSoft Integration Broker reapplies the configuration settings—including the changes—without rebooting.

Information about how to access the Gateways Properties page is discussed earlier in this chapter.

See Chapter 7, "Managing Integration Gateways," Accessing the integrationGateway.properties File, page 64.

Editing Connector Properties

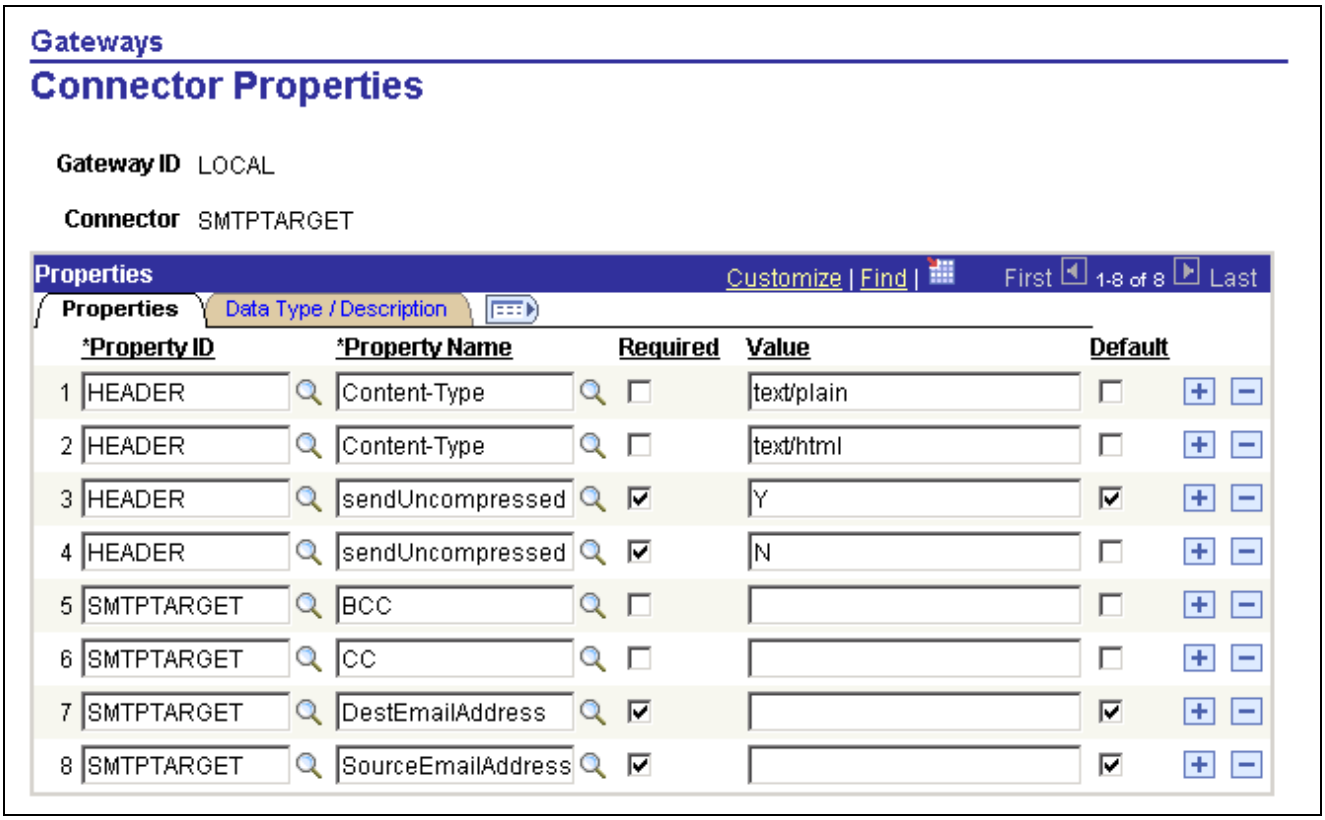
Node-level target connector properties represent parameters that can be used by the connector. These properties are hard-coded in the connector class. The Connector Properties page lists all of a connector's available properties and their values. When you specify a connector in a node definition, only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

 Copyright © 1988-2007, Oracle. All rights reserved.

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.



Properties tab for the SMTP target connector

To add a new property instance:

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

   Add a new row on the Connector Properties page.
2. Select the integration gateway with which to work. The Gateways page displays.
3. 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.
4. Select a Property ID.

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

   The available property names are specific to the property ID that you selected.
6. 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.

Copyright © 1988-2007, Oracle. All rights reserved.

7. 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.

8. (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.

9. Save the properties.
10. Click OK.

    The Gateways page appears.

Accessing Gateway Setup Properties

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

Page Used to Access Integration Gateway Properties

Page Name	Object Name	Navigation	Usage
Gateway Properties (signon) page	IBGWSIGNON	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 integration gateway properties from the Gateway component, select PeopleTools, Integration Broker, Configuration, Gateways. Click the Gateway Setup Properties link.	Sign on to access integrationGateway.properties file to set integration gateway properties.

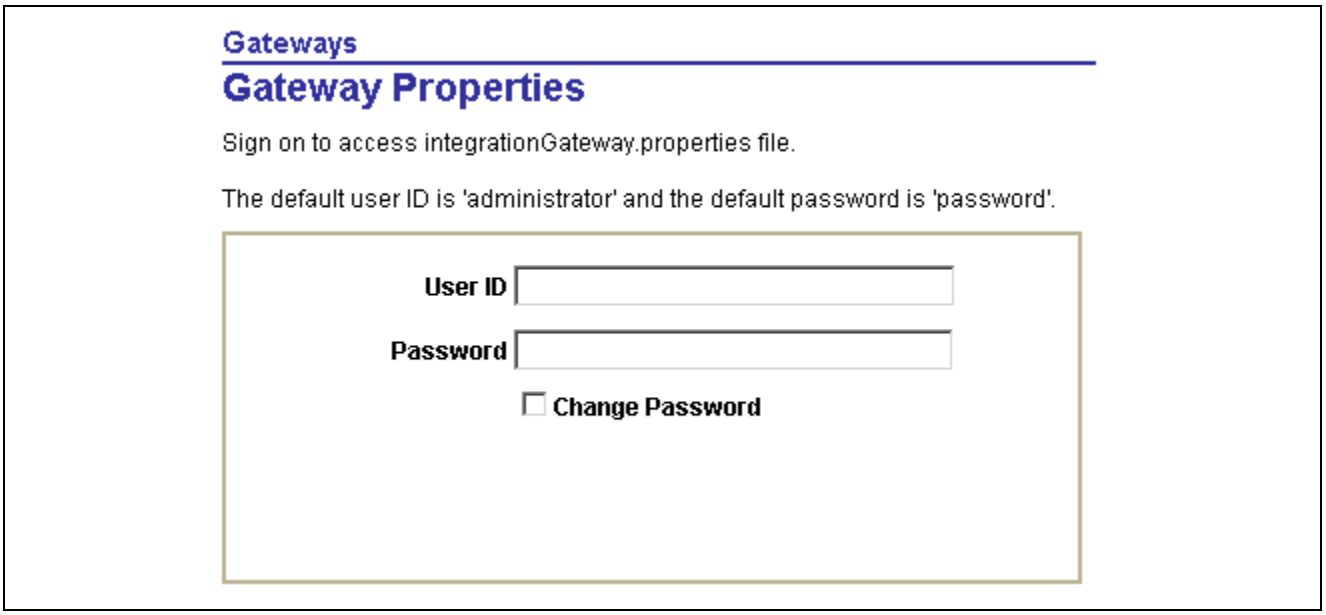
Accessing Gateway Properties

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.

 Copyright © 1988-2007, Oracle. All rights reserved.

- 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.



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 userGatewayProfile.xml file located in <PS_HOME>\webserv\<DOMAIN>\applications\peoplesoft\PSIGW\WEB-INF. The password you enter in the userGatewayProfile.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 BEA Jolt Connection Properties

The integration gateway communicates with PeopleSoft application server nodes using BEA jolt connections.

Understanding BEA Jolt Connection Properties

This section discusses setting BEA 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 provides grids for defining BEA 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.

Copyright © 1988-2007, Oracle. All rights reserved.

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 BEA 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 BEA 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.

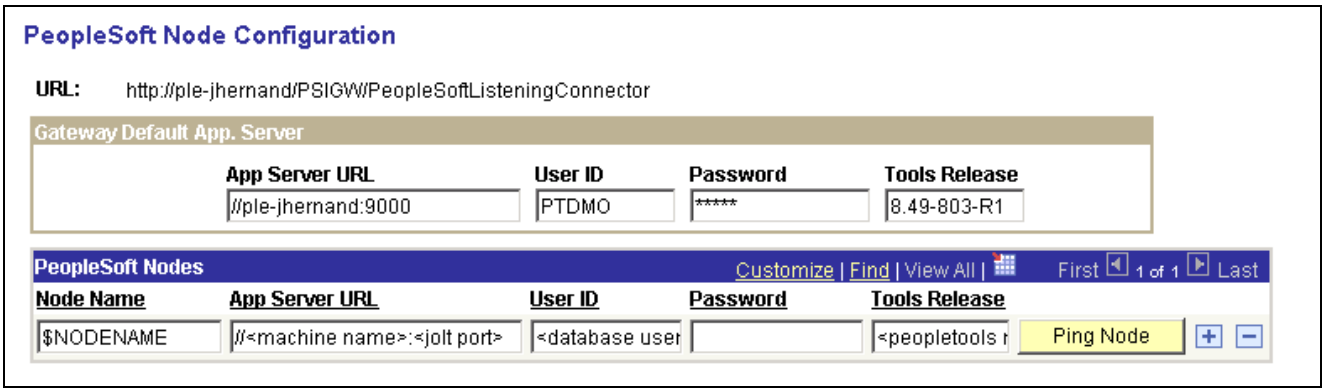
Page Used to Set BEA Jolt Connection Properties

Page Name	Object Name	Navigation	Usage
PeopleSoft Node Configuration page	PSGTWPROPS_SEC	To access the PeopleSoft Node Configuration page 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 the PeopleSoft Node Configuration page from the Gateway component, select PeopleTools, Integration Broker, Configuration, Gateways. Click the Gateway Setup Properties link.	Define BEA Jolt connection properties for unknown (default) and known nodes.

Setting BEA Jolt Connection String Properties

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

Copyright © 1988-2007, Oracle. All rights reserved.



PeopleSoft Node Configuration page

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 BEA Jolt string connection properties for unknown nodes is optional.

App Server URL(Application Server URL)

Enter the machine name and BEA 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_HOME>\appserv\<DOMAIN_NAME>.

Web Server URL

Enter the machine name and BEA 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_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, *8.49*.

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.

Copyright © 1988-2007, Oracle. All rights reserved.

See Also

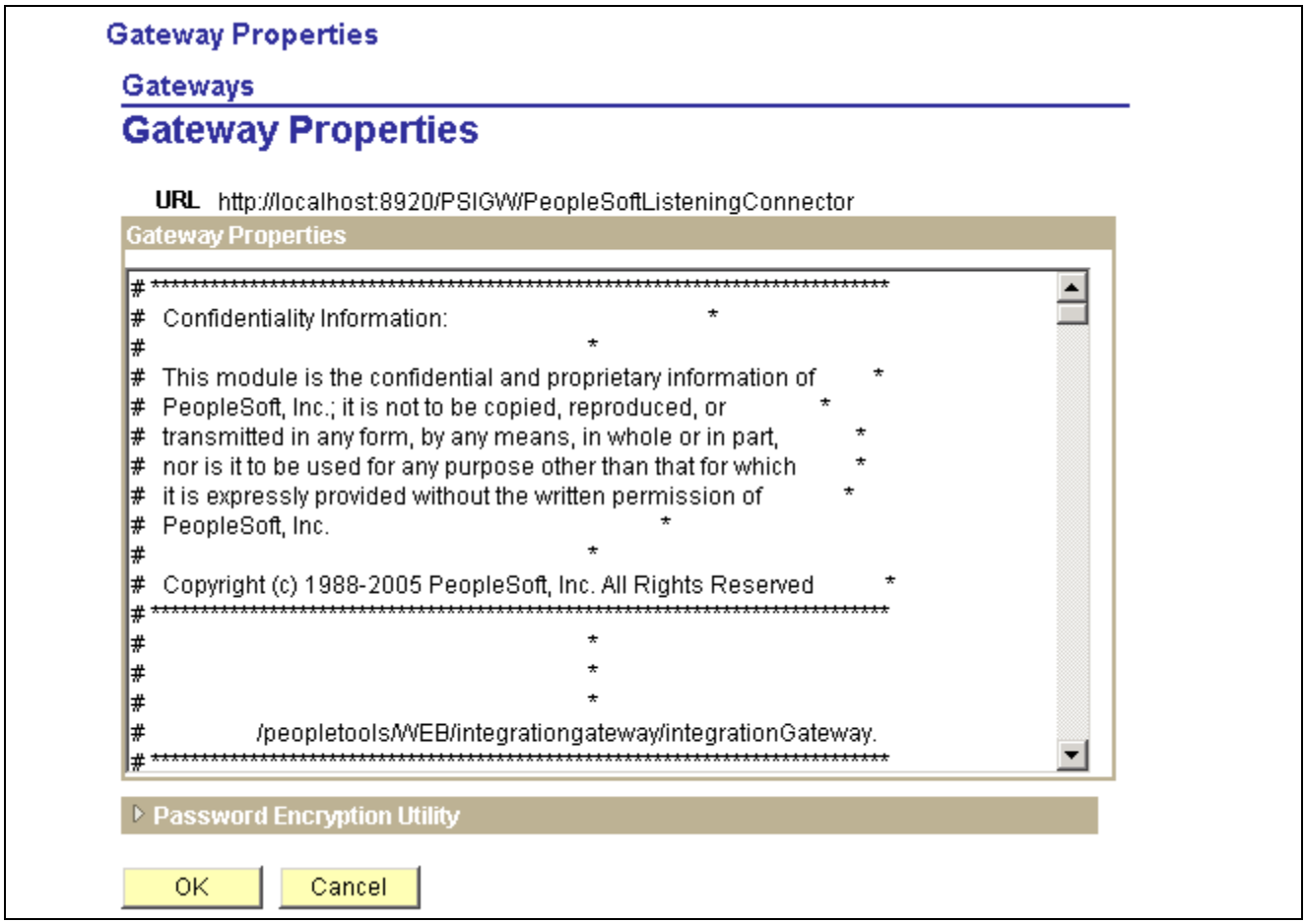
Chapter 7, “Managing Integration Gateways,” Using the integrationGateway.properties File, page 64

Chapter 7, “Managing Integration Gateways,” Configuring Security and General Properties, page 67

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 <PS_HOME> directory.

 Copyright © 1988-2007, Oracle. All rights reserved.

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 <PS_HOME> Directory

The integrationGateway.properties file is located in the following path in the PeopleSoft home directory: <PS_HOME>\webserv\<DOMAIN>\applications\peoplesoft\PSIGW\WEB-INF\integrationGateway.properties

See Also

Chapter 7, “Managing Integration Gateways,” Encrypting Passwords, page 66

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
```

Copyright © 1988-2007, Oracle. All rights reserved.

```
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.

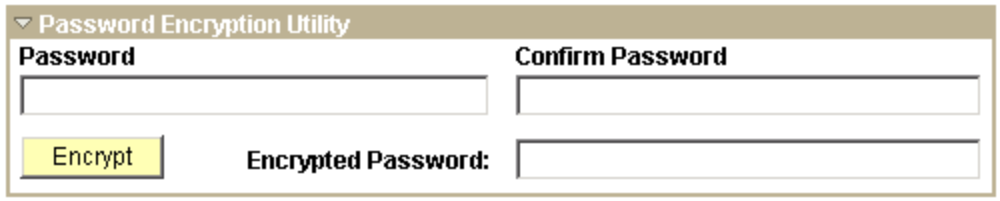
Encrypting Passwords

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 <PS_HOME> directory.

To encrypt a password:

1. Launch the PSCipher.bat file in the <PS_HOME>\webserv\peoplesoft 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.

 Copyright © 1988-2007, Oracle. All rights reserved.

- On a Windows NT or Windows 2000 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 security properties.
- Specify the gateway version.
- Specify the gateway class location.
- Set general connection properties.
- Set logging properties.
- Set DTD validation properties.
- Set BEA Jolt session pooling parameters.

Understanding Integration Gateway Properties and OAS Clustering

If using OAS web server and implementing clustering, you must specify the path to the cluster component for the following integration gateway properties:

- ig.messageLog.filename
- ig.errorLog.filename
- secureFileKeystorePath

For example:

```
ig.messageLog.filename=<OAS_HOME>/j2ee/<component_name>/applications/<app_name>
/PSIGW/msgLog.html
```

A red paper posted on Customer Connection titled *Clustering and High Availability for Enterprise PeopleTools 8.4x* provides additional information. See chapter 3, “Configuring an Oracle Application Server Cluster with PeopleTools 8.47”

See http://www.peoplesoft.com/media/cupa/pdf/red_paper/clustering__8_4.pdf

Setting 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.

Copyright © 1988-2007, Oracle. All rights reserved.

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*.

You must set the following properties in integrationGateway.properties so that the gateway can access the previously installed 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 7, "Managing Integration Gateways," Encrypting Passwords, page 66.
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 <PS_HOME>\webserv\<DOMAIN>\keystore
secureFileKeystorePasswd	Enter the keystore password, which is typically the default, *password.* This password should not be encrypted.

See Also

Chapter 27, "Setting Up Secure Integration Environments," page 583

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 8.49 integration gateway to communicate with a PeopleTools 8.49 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 8.49 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, *8.49*.

 Copyright © 1988-2007, Oracle. All rights reserved.

Specifying the Gateway Class Location

This required property indicates where the integration gateway classes are installed. You can find it in the section of the integrationGateway.properties file labeled *Integration Gateway CLASS INSTALLATION Section*. Specify the location as follows:

```
ig.installdir=directory_path
```

where *directory_path* is the location of the gateway Java classes in the web server directory structure. This is typically <PS_HOME>\webserv\<DOMAIN>\applications\peoplesoft\PSIGW\WEB-INF\classes.

Note. If you deploy the PeopleSoft Pure Internet Architecture installation through an Enterprise Archive (EAR) file on a different machine under a different directory, the ig.installdir value you specify here will be invalid. The program will still be functional because in instances like this, the path is built based on the class file location.

Setting General Connection Properties

This section discusses:

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

The general connection properties include default connector properties and BEA 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.

Copyright © 1988-2007, Oracle. All rights reserved.

Property	Description
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 BEA 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
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
ig.isc.password=VOBN5KcQZMg
ig.isc.toolsRel=8.49
```

See Chapter 7, "Managing Integration Gateways," Setting General Connection Properties, page 69.

BEA Jolt Connect String Properties for Known Nodes

You must set four BEA 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 BEA 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.$NODENAME.serverURL	Enter the URL of the application server node, consisting of the machine name and BEA Jolt port; for example: `ig.isc.MYNODE.serverURL=//MYMACHINE:9000` **Note.** You can determine the Jolt port of the application server by examining the *JOLT Listener* section in the psappsrv.cfg file located in <PS_HOME>\appserv\<DOMAIN_NAME>.
ig.isc.$NODENAME.userid	Enter the UserID that you defined when you created the application server domain; for example: `ig.isc.MYNODE.userid=TOPDOG`
ig.isc.$NODENAME.password	Enter UserPswd that you defined when you created the application server domain. This password must be encrypted; for example: `ig.isc.MYNODE.password=VOBN5KcQZMg` See Chapter 7, "Managing Integration Gateways," Encrypting Passwords, page 66.
ig.isc.$NODENAME.toolsRel	Enter the version number of PeopleTools installed on the application server node to two decimal places; for example: `ig.isc.MYNODE.toolsrel=8.49`

Copyright © 1988-2007, Oracle. All rights reserved.

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 JPEG image to use when displaying error and message log documents. The default location and image name PSbackground.jpg. By default it is located in <PS_HOME>\webserv\<DOMAIN> \applications\peoplesoft\PSIGW. 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.

Copyright © 1988-2007, Oracle. All rights reserved.

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 <PS_HOME>\webserv\<DOMAIN> \applications\peoplesoft \PSIGW\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.
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 <PS_HOME>\webserv\<DOMAIN> \applications\peoplesoft \PSIGW\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

Chapter 22, “Managing Error Handling, Logging, Tracing, and Debugging,” page 497

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.

Copyright © 1988-2007, Oracle. All rights reserved.

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 BEA 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*.

Applying Message Transformations at the Integration Gateway

Typically, you apply filtering, transformation, and data translation to a message at the node level on the application server by using a transaction modifier to invoke an Application Engine transform program. However, on systems with high transaction volumes, Application Engine transformations can constrict message throughput. To improve performance, you can apply XSLT transformation programs at an integration gateway.

Note. While you may apply transformations at the integration gateway level, PeopleSoft strongly recommends that you apply them at the application server level due to a more robust infrastructure to support them.

See http://www.apache.com

See Also

Chapter 20, “Applying Filtering, Transformation and Translation,” page 373

Understanding Applying Message Transformations at the Integration Gateway

Only XSLT transformations can be applied at the gateway. Message filtering, data translation, and PeopleCode transformations must still be applied at the node using an Application Engine transform program, and can be applied in addition to gateway-based transformations.

You can apply XSLT transformations at any gateway that handles the message you want to transform.

When a gateway with transformation enabled processes an IBRequest, it examines the transformation properties in the integrationGateway.properties file to determine if they specify a transformation for the same message with the same source and target nodes as the IBRequest. If these values match, the gateway compiles the specified XSLT transformation program and applies it to the message, then sends the transformed message to the target node.

 Copyright © 1988-2007, Oracle. All rights reserved.

Note. The IBRequest can specify only a RequestingNode or only a DestinationNode, but it must specify at least one of these values—ig.DefaultServer.LocalNode supplies the other one.

With synchronous transactions, the gateway applies transformations only to the request message, not to the response message. If the original message is compressed and base64 encoded, the gateway decompresses and decodes it before applying the transformation, then compresses and encodes it again before sending.

Note. The integration gateway retains all compiled XSLT transformation programs in a memory cache to improve performance during subsequent transformations. If you edit the code of a transformation program that's been used before, you must purge the compiled programs from the cache so the new version will be recompiled. To do this, click the Refresh button on the gateway definition.

Developing and Implementing Gateway-Based Transformation Programs

Developing and implementing gateway-based transform programs requires the following activities:

1. Determine if the message you want to transform qualifies for gateway-based transformation:
   - The message content must be XML-DOM compliant.
   - The message must not have nonrepudiation activated.
2. Develop the XSLT transformation program.

   See Chapter 20, "Applying Filtering, Transformation and Translation," Applying Transformations, page 397.

   You can develop, test and debug the program within Application Engine, but you must save the program code as an external text file. Place the file in any location that can be accessed from the integration gateway machine, for example:

   ```
   C:\XSLProgs\MyTransform.xsl
   ```
3. Configure the appropriate gateway property settings in the integrationGateway.properties file to enable the transformation.

   See Chapter 7, "Managing Integration Gateways," Setting Integration Gateway Properties for Gateway-Based Transformations, page 75.
4. Refresh the gateway properties.

Setting Integration Gateway Properties for Gateway-Based Transformations

To apply gateway-based transformations, set the following properties in the integrationGateway.properties file.

For each message you want to transform, you must create a set of property entries using the same number, which associate a given transformation program with that message. However, you can specify the same transformation program for multiple messages.

When entering these settings, each transformation must be numbered for identification, using the convention ig.transform1, ig.transform2, ig.transform3, and so on.

Copyright © 1988-2007, Oracle. All rights reserved.

Property	Description
ig.isGatewayTransformationEnabled	Specify whether transformation is enabled for this gateway. Valid values are: • *TRUE*. Transformation is enabled. • *FALSE*. Transformation is disabled — the integration gateway will ignore the other transformation properties. This is the default value.
ig.DefaultServer.LocalNode	Enter the name of the node definition that will be used as the source or destination node for a given transformation if either of those values isn't identified; for example you must specify `ig.DefaultServer.LocalNode=DEF_NODE` All transformations require that you specify both a source node and a destination node. This property applies if either the ig.transformN.SourceNode property or the ig.transformN.DestinationNode property is empty or invalid, or if the IBRequest doesn't specify either RequestingNode or DestinationNode.
ig.transforms	Specify the number of transformations configured in the integrationGateway.properties file; for example: `ig.transforms=7`

ig.transform*N*.XSL	Enter the full path and filename of transformation program *N*.
	Your path specification must use either double back slashes or single forward slashes as separators; for example
	`ig.transform4.XSL=C:\\XSLProgs\\MyTransform.xsl`
	`ig.transform4.XSL=C:/XSLProgs/MyTransform.xsl`
	`ig.transform4.XSL=/usr/xsls/MyTransform.xsl`

ig.transform*N*.MessageName	Enter the name of the message to be transformed by transformation program N; for example: `ig.transform4.MessageName=MY_MSG_A`
ig.transform*N*.SourceNode	Enter the name of the source node from which the original message is being sent, or enter the value *ANY*; for example: `ig.transform4.SourceNode=NODE_Aig.transform4.SourceNode=ANY` If this value is *ANY*, the value of the ig.DefaultServer.LocalNode property will be used instead.

Copyright © 1988-2007, Oracle. All rights reserved.

Property	Description
ig.transform*N*.DestinationNode	Enter the name of the target node to which the transformed message is being sent, or enter the value ANY; for example: `ig.transform4.DestinationNode=NODE_Big.transform4.DestinationNode=ANY` If this value is *ANY*, the value of the ig.DefaultServer.LocalNode property will be used instead.
ig.transform*N*.DestinationMessageName	(Optional.) Enter the name that the target node uses for the transformed version of the message, if it's different from the original message name; for example: `ig.transform4.DestinationMessageName=MY_MSG_B` This enables the gateway to rename the message before sending it, so the target node will recognize and accept it.

Understanding Logged Errors

If an error occurs when you refresh the gateway properties or during a transformation, it's entered in the gateway's error log file.

Integration Gateway Refresh Errors

After you specify integration gateway properties and refresh the gateway, errors can be generated for the following reasons:

- No value is specified for ig.transform*N*.XSL.
- No value is specified for ig.transform*N*.MessageName.
- No value is specified for both ig.DefaultServer.LocalNode and ig.transform*N*.SourceNode.
- No value is specified for both ig.DefaultServer.LocalNode and ig.transform*N*.DestinationNode.
- The gateway is in the process of loading, compiling or caching a transformation program.

Runtime Transformation Errors

Errors are generated for the following reasons when the gateway attempts to apply a transformation:

- Nonrepudiation is enabled for the message.
- The integration gateway is unable to transform the message.
- The integration gateway is unable to decompress and decode the message.
- The integration gateway is unable to compress and encode the message.
- The IBRequest does not specify a RequestingNode and no value is specified for ig.DefaultServer.LocalNode.
- The IBRequest does not specify a DestinationNode and no value is specified for ig.DefaultServer.LocalNode.
- The IBRequest specifies neither a RequestingNode nor a DestinationNode.

Copyright © 1988-2007, Oracle. All rights reserved.

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

Chapter 12, "Sending and Receiving Messages," Generating and Sending Messages, page 221

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.

 Copyright © 1988-2007, Oracle. All rights reserved.

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");
```

Copyright © 1988-2007, Oracle. All rights reserved.

 Copyright © 1988-2007, Oracle. All rights reserved.

CHAPTER 8

Understanding Supported Message Structures

This chapter discusses the internal format PeopleSoft uses to exchange request and response messages between the integration gateway and the application server. This chapter discusses:

- Internal message format for request messages.
- Internal message format for response messages.
- Accessing IBInfo elements using PeopleCode.
- Rowset-based message structure.
- PSCAMA.
- Identifying changes to field-level attributes.
- Nonrowset-based message structures.
- XML DOM-Compliant messages.
- SOAP-Compliant messages.
- Non-XML messages.
- Message part structures.
- Message container structures.

Integration Broker Message Structures

This section discusses the internal message formats for request messages and response messages, local compression, and how to access IBInfo elements.

Internal Message Format for Request Messages

This section discusses the format used to exchange request messages between the integration gateway and the application server. These messages are frequently referred to as IBRequest messages.

The Multipurpose Internet Mail Extension standard (MIME) is used as the basic structure for internal messaging. MIME has several advantages in that the standard is well-known and supported, and because it is text-based, it is human readable and easily serializable.

Messages using the internal format display in the integration gateway log file. Since the log file is a valuable tool for debugging, anyone reading the file will need to understand how the messages are structured.

Every request message contains three parts:

Copyright © 1988-2007, Oracle. All rights reserved.

Headers	The first part of a request message contains headers which describe the attributes of the whole message.
IBInfo Integration Broker Information	The IBInfo section contains the credentials of the request as well as all other information required by the PeopleSoft Integration Broker to process the message. The IBInfo for a request has a specific XML structure which is used for all request messages in the system, regardless if the message is being sent to the application server or to the integration gateway.
Content section	The final section contains the message body of the original request. This is the payload and is what is ultimately delivered to the final destination.

The following is an example of a request message in the PeopleSoft internal MIME format:

```
Message-ID: <-123.123.123.123@nowhere >
Mime-Version: 1.0
Content-Type: multipart/related; boundary="Integration_Server_MIME_Boundary"
Content-ID: PeopleSoft-Internal-Mime-Message
PeopleSoft-ToolsRelease: 8.49

--Integration_Server_MIME_Boundary
Content-Type: text/plain; charset=UTF-8
Content-Transfer-Encoding: 8bit
Content-ID: IBInfo
Content-Disposition: inline

<?xml version="1.0" ?>
  <IBInfo>
   <TransactionID>
     <![CDATA[ caa3a040-bde5-11da-914c-ecaede80d83b]]>
  </TransactionID>
  <ExternalOperationName>
  <![CDATA[ QE_FLIGHTPLAN_TRANSFORM.VERSION_1]]>
  </ExternalOperationName>
  <OperationType>async</OperationType>
  <From>
    <RequestingNode>
       <![CDATA[ QE_LOCAL]]>
    </RequestingNode>
    <RequestingNodeDescription>
       <![CDATA[ ]]>
    </RequestingNodeDescription>
    <NodePassword>
       <![CDATA[ password]]>
    </NodePassword>
    <ExternalUserName>
    <![CDATA[ ]]>
    </ExternalUserName>
    <ExternalUserPassword>
    <![CDATA[ ]]>
    </ExternalUserPassword>
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
<AuthToken>
   <![CDATA[ owAAAAQDAgEBAAAAvAIAAAAAAAAsAAAABABTaGRyAk4AbQg4AC4AMQ
    AwABTFZOonLEjJaPtR6v02oadvRUoSq2MAAAAFAFNkYXRhV3icHYhNDkAwGERfEQ
    srFyFN0cZSaGz8xAmcwA0dzug3yZv53gMUeWaM+s1IV11EFnZOysjBSv2bm01mZl
    L3Dqt4GrETHSHtQCs6cWBM2ybr9fMBbP0LSQ==]]>
</AuthToken>
<WSA-ReplyTo>
   <![CDATA[ ]]>
</WSA-ReplyTo>
<NodeDN>
   <![CDATA[ ]]>
</NodeDN>
<OrigUser>
   <![CDATA[ QEDMO]]>
</OrigUser>
<OrigNode>
   <![CDATA[ QE_LOCAL]]>
</OrigNode>
<OrigProcess>
   <![CDATA[ QE_FLIGHTDATA]]>
</OrigProcess>
<OrigTimeStamp>2006-03-27T15:02:39.280000-0800</OrigTimeStamp>
<DirectGatewayRequest />
<SyncServiceTimeout />
<ExternalMessageID>
   <![CDATA[ ]]>
</ExternalMessageID>
<SegmentsUnOrder>N</SegmentsUnOrder>
<ConversationID>
   <![CDATA[ ]]>
</ConversationID>
<WSA-MessageID>
   <![CDATA[ ]]>
</WSA-MessageID>
<InReplyToID>
   <![CDATA[ ]]>
</InReplyToID>
<DataChunk>
   <![CDATA[ ]]>
</DataChunk>
<DataChunkCount>
   <![CDATA[ ]]>
</DataChunkCount>
</From>
<WS-Security>
  <WSTokenType>
     <![CDATA[ ]]>
  </WSTokenType>
</WS-Security>
<To>
```

Copyright © 1988-2007, Oracle. All rights reserved.

```
  <DestinationNode>
      <![CDATA[ QE_IBTGT]]>
  </DestinationNode>
  <FinalDestinationNode>
     <![CDATA[ ]]>
  </FinalDestinationNode>
  <AppServerDomain>
     <![CDATA[ ]]>
  </AppServerDomain>
</To>
<Cookies>
   <![CDATA[ ]]>
</Cookies>
<PathInfo>
   <![CDATA[ ]]>
</PathInfo>
<HttpSession>
  <SessionID>
     <![CDATA[ ]]>
  </SessionID>
</HttpSession>
<QStrArgs />
<ContentSections>
   <ContentSection>
      <ID>ContentSection0</ID>
      <NonRepudiation>N</NonRepudiation>
      <Headers>
        <version>
           <![CDATA[ VERSION_1]]>
        </version>
        <encoding>
           <![CDATA[ base64(deflate)]]>
        </encoding>
        <encodedlength>
           <![CDATA[ 948(709)]]>
        </encodedlength>
        <length>
           <![CDATA[ 2840]]>
        </length>
      </Headers>
   </ContentSection>
 </ContentSections>
<PublishNode>
   <![CDATA[ QE_LOCAL]]>
</PublishNode>
<Queue>
   <![CDATA[ QE_FLIGHTPLAN_CHNL]]>
</Queue>
<SubQueue>
   <![CDATA[ c9de8110-bde5-11da-be79-846b4e717ebf]]>
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
      </SubQueue>
      <VisitedNodes>
         <![CDATA[ QE_LOCAL|]]>
      </VisitedNodes>
      <Connector>
         <ConnectorName>
            <![CDATA[ PSFTTARGET]]>
         </ConnectorName>
         <ConnectorClassName>
            <![CDATA[ PeopleSoftTargetConnector]]>
         </ConnectorClassName>
         <RemoteFrameworkURL>
            <![CDATA[ ]]>
         </RemoteFrameworkURL>
         <ConnectorParameters>
            <ConnectorParam>
               <Name>
                 <![CDATA[ URL]]>
               </Name>
               <Value>
                  <![CDATA[ ]]>
               </Value>
            </ConnectorParam>
         </ConnectorParameters>
        <ConnectorHeaders />
    </Connector>
   <AttachmentSection ResponseAsAttachment="N" />
</IBInfo>

--Integration_Server_MIME_Boundary
Content-Type: text/plain; charset=UTF-8
Content-Transfer-Encoding: 8bit
Content-ID: ContentSection0
Content-Disposition: inline

eJydll1P2zAUhu8n7T+g/gAKFdt6UYxOHCe1FH9gO1RwE1UMTUiMTRRN27+fk7SpvwItd/F57c
TvG+c5WVz9/fl08ufhZfP46/lycn56NrlCnz8trklTVLRcGlkBb4wCrguhGFoUjw9P382/3w8b
tJ+Ug4GT+6f1ZnM5UZNOAMxrlhF18mrnXk76wWTaaUzTRhOuxU7FS9hpopBxEbC5lSSuY6gqTU
seK/q6hlyJhIIFY+fp8iwuE8yCYk40VpXgZVCfeml0i1eSN1IRTYw+Ih1twFDBG569KUuhU/KK
gBQ8HVdR37VGeUHDzW9FIdtbx9oK1JJAnnDsWOxmcrihZbfBIxyv4FYKys2Y5YyAolHgVrDHsk
z4hMpQU+cJpYJRxb7REam1nnz8DakEpoYSHWs2iPakJCLbR7T1kIPaZnhEZv0yTSqCnXcWT2EiZ
alfLSQZ0TNQGkPiJHQLR2I3pYFU3V5yTqrWX/yq7iirzTIWpCoS2blh9esVA0b4Mcm9831BORIJ
TWy//9q0JDg8AlNnc2ghbZrMQ6TFalnbuBocflZQ59S0yAvjz0C3J3hsHdOlPQ/XFkLhUZVKYKJ
1Q7n1zjGJbMs6q6heNqquSEMTN+Y2Em69hCZ7X/bCbQts8yNfv67Rwrysnzfr+1fbWg5rFmj+bT
7ro9tV/H6B7C1UN8GpbdsGmp9eXHRaO9i3DVScz7f37IZue0Bfv1z0DxwqQ49AGXRKPxh6BMrwU
J7tegSyiRRduR3se8TAgrehiBKmXSh6GHTQ5+POR1yANQ9lIb48ZH0YUykXAaYCMKVg5AEohI4L
mhAuHlAGiHwQHCkvDjhcVAx4iJAwPfAv4KCHOX0/7PRRdw87utfFU53bp1X4K/MmvRDh5WLqFhy
CJajVz4+qLr4SyEJnFjZhLeSWzyqPTx6KpgOh9k6D/9z/1gQ1Ww==
--Integration_Server_MIME_Boundary--
```

Copyright © 1988-2007, Oracle. All rights reserved.

IBRequest Header Section

The first part of a request message contains headers which describe the attributes of the whole message.

```
Message-ID: <-123.123.123.123@nowhere >
Mime-Version: 1.0
Content-Type: multipart/related; boundary="Integration_Server_MIME_Boundary"
Content-ID: PeopleSoft-Internal-Mime-Message
PeopleSoft-ToolsRelease: 8.49

--Integration_Server_MIME_Boundary
Content-Type: text/plain; charset=UTF-8
Content-Transfer-Encoding: 8bit
Content-ID: IBInfo
Content-Disposition: inline
```

IBRequest IBInfo Section

The following example shows an IBInfo section for a request message that was sent from the application server to the integration gateway (formatted for easier reading):

```
<?xml version="1.0" ?>
  <IBInfo>
   <TransactionID>
     <![CDATA[ caa3a040-bde5-11da-914c-ecaede80d83b]]>
  </TransactionID>
  <ExternalOperationName>
  <![CDATA[ QE_FLIGHTPLAN_TRANSFORM.VERSION_1]]>
  </ExternalOperationName>
  <OperationType>async</OperationType>
  <From>
    <RequestingNode>
       <![CDATA[ QE_LOCAL]]>
    </RequestingNode>
    <RequestingNodeDescription>
       <![CDATA[ ]]>
    </RequestingNodeDescription>
    <NodePassword>
       <![CDATA[ password]]>
    </NodePassword>
    <ExternalUserName>
    <![CDATA[ ]]>
    </ExternalUserName>
    <ExternalUserPassword>
    <![CDATA[ ]]>
    </ExternalUserPassword>
    <AuthToken>
       <![CDATA[ owAAAAQDAgEBAAAAvAIAAAAAAAAsAAAABABTaGRyAk4AbQg4AC4AMQ
        AwABTFZOonLEjJaPtR6v02oadvRUoSq2MAAAAFAFNkYXRhV3icHYhNDkAwGERfEQ
        srFyFN0cZSaGz8xAmcwA0dzug3yZv53gMUeWaM+s1IV11EFnZOysjBSv2bm01mZl
        L3Dqt4GrETHSHtQCs6cWBM2ybr9fMBbP0LSQ==]]>
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
</AuthToken>
<WSA-ReplyTo>
   <![CDATA[ ]]>
</WSA-ReplyTo>
<NodeDN>
   <![CDATA[ ]]>
</NodeDN>
<OrigUser>
   <![CDATA[ QEDMO]]>
</OrigUser>
<OrigNode>
   <![CDATA[ QE_LOCAL]]>
</OrigNode>
<OrigProcess>
   <![CDATA[ QE_FLIGHTDATA]]>
</OrigProcess>
<OrigTimeStamp>2006-03-27T15:02:39.280000-0800</OrigTimeStamp>
<DirectGatewayRequest />
<SyncServiceTimeout />
<ExternalMessageID>
   <![CDATA[ ]]>
</ExternalMessageID>
<SegmentsUnOrder>N
</SegmentsUnOrder>
<ConversationID>
   <![CDATA[ ]]>
</ConversationID>
<WSA-MessageID>
   <![CDATA[ ]]>
</WSA-MessageID>
<InReplyToID>
   <![CDATA[ ]]>
</InReplyToID>
<DataChunk>
   <![CDATA[ ]]>
</DataChunk>
<DataChunkCount>
   <![CDATA[ ]]>
</DataChunkCount>
</From>
<WS-Security>
  <WSTokenType>
     <![CDATA[ ]]>
  </WSTokenType>
</WS-Security>
<To>
  <DestinationNode>
      <![CDATA[ QE_IBTGT]]>
  </DestinationNode>
  <FinalDestinationNode>
```

Copyright © 1988-2007, Oracle. All rights reserved.

```
      <![CDATA[ ]]>
  </FinalDestinationNode>
  <AppServerDomain>
     <![CDATA[ ]]>
  </AppServerDomain>
</To>
<Cookies>
   <![CDATA[ ]]>
</Cookies>
<PathInfo>
   <![CDATA[ ]]>
</PathInfo>
<HttpSession>
  <SessionID>
     <![CDATA[ ]]>
  </SessionID>
</HttpSession>
<QStrArgs />
<ContentSections>
   <ContentSection>
      <ID>ContentSection0</ID>
      <NonRepudiation>N</NonRepudiation>
      <Headers>
        <version>
           <![CDATA[ VERSION_1]]>
        </version>
        <encoding>
           <![CDATA[ base64(deflate)]]>
        </encoding>
        <encodedlength>
           <![CDATA[ 948(709)]]>
        </encodedlength>
        <length>
           <![CDATA[ 2840]]>
        </length>
      </Headers>
   </ContentSection>
 </ContentSections>
<PublishNode>
   <![CDATA[ QE_LOCAL]]>
</PublishNode>
<Queue>
   <![CDATA[ QE_FLIGHTPLAN_CHNL]]>
</Queue>
<SubQueue>
   <![CDATA[ c9de8110-bde5-11da-be79-846b4e717ebf]]>
</SubQueue>
<VisitedNodes>
   <![CDATA[ QE_LOCAL|]]>
</VisitedNodes>
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
      <Connector>
         <ConnectorName>
            <![CDATA[ PSFTTARGET]]>
         </ConnectorName>
         <ConnectorClassName>
            <![CDATA[ PeopleSoftTargetConnector]]>
         </ConnectorClassName>
         <RemoteFrameworkURL>
            <![CDATA[ ]]>
         </RemoteFrameworkURL>
         <ConnectorParameters>
            <ConnectorParam>
               <Name>
                 <![CDATA[ URL]]>
               </Name>
               <Value>
                  <![CDATA[ ]]>
               </Value>
            </ConnectorParam>
         </ConnectorParameters>
        <ConnectorHeaders />
     </Connector>
    <AttachmentSection ResponseAsAttachment="N" />
  </IBInfo>
```

While the basic structure is the same for all requests, not all elements are always required. An example of this is the Connector section. The Connector XML is used to tell the integration gateway to route a message to the named target connector. It also lists configuration parameters for the outbound request. This XML would only be seen in requests sent from the application server to the integration gateway. For requests going in the other direction, the section would be empty.

Note. The only element that is always required is *ExternalOperationName*.

The following is a list of the most important elements that may appear in the IBInfo section:

Element	Description
IBInfo / ExternalOperationName	The name of the requested service operation.
IBInfo / Operation Type	(Optional.) This is the type of service operation. The valid values are: asynchronous, synchronous and ping.
IBInfo / TransactionID	(Optional.) The transaction ID is used to uniquely identify a request.
IBInfo / From / RequestingNode	(Optional.) The requesting node is the node that sent the request to the current system.
IBInfo / From / Password	(Optional.) This is the password for the requesting node.
IBInfo / From / DN	(Optional.) For incoming requests, the DN gives the Distinguished Name extracted from the certificate authentication process.

Copyright © 1988-2007, Oracle. All rights reserved.

Element	Description
IBInfo / From / OrigNode	(Optional.) For requests that cross multiple nodes, OrigNode is used to identify the node that initiated the request.
IBInfo / From / OrigTimeStamp	(Optional.) This timestamp corresponds to the time that the request was created. For requests that cross nodes, this is the time that the first request was created.
IBInfo / To / DestinationNode	(Optional.) This is the node to which the request will be delivered.
IBInfo / To / FinalDestinationNode	(Optional.) In cases where the message will be passed across several nodes, this value specifies the ultimate target of the message.
IBInfo / QStrArgs	(Optional.) Specific to incoming HTTP requests. These are the query string parameters found when the request was received by the HTTP listening connector.
IBInfo / Cookies	(Optional.) Specific to incoming HTTP requests. This is cookie string found when the request was received by the HTTP listening connector.
IBInfo / PathInfo	(Optional.) Specific to incoming HTTP requests. This is the path information extracted from the request.
IBInfo / ContentSections / ContentSection	(Optional.) This node provides metadata about the text present in the ContentSection.
IBInfo / ContentSections / ContentSection / ID	(Optional.) The index number of the content section.
IBInfo / ContentSections / ContentSection / NonRepudiation	(Optional.) Indicates as to whether nonrepudiation should be performed.
IBInfo / ContentSections / ContentSection / Headers	(Optional.) Provides additional information about the data.
IBInfo / PublishingNode	(Optional.) The node that published the message.
IBInfo / Queue	(Optional.) The queue to which the service operation was published.
IBInfo / InternalInfo / AppMsg / SubQueue	(Optional.) The subqueue to which the service operation was published.
IBInfo / InternalInfo / AppMsg / VisitedNodes	(Optional.) The list of nodes that have already received this message. This is useful when a message is being propagated across multiple nodes.
IBInfo / InternalInfo / AppMsg / PublicationID	(Optional.) The publication ID for this message.
IBInfo / Connector	(Optional.) Connector information instructs the gateway as to how to process the request.
IBInfo / Connector / ConnectorName	(Optional.) This is the proper name of the target connector to invoke to send the message.
IBInfo / Connector / ConnectorClassName	(Optional.) This is the class name of the target connector to invoke.

Copyright © 1988-2007, Oracle. All rights reserved.

Element	Description
IBInfo / Connector / ConnectorParameters	(Optional.) Connector parameters are processing instructions for the target connector to be invoked.
IBInfo / Connector / ConnectorHeaders	(Optional.) Connector headers provide further metadata about the contents of the message to be sent.

IBRequest Content Section

The content section of a request message features the message body.

```
?xml version="1.0"?>
<TestXml>This is a sample request message.</TestXml>
```

Internal Message Format for Response Messages

The internal format for response messages parallels that for request messages, and has the same basic MIME structure. These messages are frequently referred to as IBResponse messages.

There are three logical components to a MIME response message: the IBResponse header section, the IBInfo section, and the Content section.

The following code shows an example of a response message:

```
Message-ID: <32004392.1143500580241.JavaMail.KCOLLIN2@PLE-KCOLLIN2>
Date: Mon, 27 Mar 2006 15:03:00 -0800 (PST)
Mime-Version: 1.0
Content-Type: multipart/related;
 boundary="----=_Part_4_9069393.1143500580221"
Content-ID: PeopleSoft-Integration-Broker-Internal-Mime-Message
PeopleSoft-ToolsRelease: 8.49

------=_Part_4_9069393.1143500580221
Content-Type: text/plain; charset=UTF-8
Content-Transfer-Encoding: 8bit
Content-Disposition: inline
Content-ID: IBInfo

<?xml version="1.0"?><IBInfo><Status><StatusCode>0</StatusCode>
<MsgSet>158</MsgSet>

<MsgID>10000</MsgID><DefaultTitle>Integration Broker Response
Message</DefaultTitle>

</Status><ContentSections><ContentSection><ID>ContentSection0</ID>
<NonRepudiation>N</NonRepudiation></ContentSection></ContentSections></IBInfo>
------=_Part_4_7210339.1008355101202
```

IBResponse Header

The first part of a response message contains headers which describe the attributes of the whole message.

Copyright © 1988-2007, Oracle. All rights reserved.

```
Message-ID: <32004392.1143500580241.JavaMail.KCOLLIN2@PLE-KCOLLIN2>
Date: Mon, 27 Mar 2006 15:03:00 -0800 (PST)
Mime-Version: 1.0
Content-Type: multipart/related;
 boundary="----=_Part_4_9069393.1143500580221"
Content-ID: PeopleSoft-Integration-Broker-Internal-Mime-Message
PeopleSoft-ToolsRelease: 8.49
```

IBResponse IBInfo Section

The format for the XML for the IBInfo for a response message is different than that for the request message. The following is a sample (formatted for easier reading):

```
<?xml version="1.0"?>
<IBInfo>
   <Status>
      <StatusCode>0</StatusCode>
      <MsgSet>158</MsgSet>
      <MsgID>10000</MsgID>
      <DefaultMsg>OK</DefaultMsg>
      <DefaultTitle>Integration Broker Response Message</DefaultTitle>
   </Status>
   <ContentSections>
      <ContentSection>
      <ID>ContentSection0</ID>
      <NonRepudiation>N</NonRepudiation>
      </ContentSection>
   </ContentSections>
</IBInfo>
```

The following is the list of all the elements that may be present in the IBInfo for a response:

Element	Description
IBInfo / Status / StatusCode	Describes the result of the request. The possible values are: • *0* (zero). Request successfully processed. • *10*. Temporary error occurred. Request can be resent. • *20*. Fatal error occurred. Do not resend request. • *30*. Request message is a duplicate of a message previously received.
IBInfo / Status / MsgSet	The MessageSetNumber for this message in the Message Catalog. Message set number 158 is assigned to the PeopleSoft Integration Broker.
IBInfo / Status / MsgID	The Message Number for this message in the Message Catalog. If no errors occurred during the processing of the request, the MsgID will be set to the value ‘10000’.
IBInfo / Status / DefaultTitle	Used if the message catalog is unavailable. This value corresponds to the “Message Text” for a given entry in the message catalog.

Copyright © 1988-2007, Oracle. All rights reserved.

Element	Description
IBInfo / Status / DefaultMsg	Used if the message catalog is unavailable. This value corresponds to the "Explanation" for a given entry in the message catalog.
IBInfo / Status / Parameters	Parameters may be used to provide additional information for error responses.
IBInfo / ContentSection	A description of the content section returned with the response. **Note.** Not all response messages will have a content section. The structure of the content section and all child elements is the same as was seen in the request IBInfo.

IBResponse Content Section

The content section of a response message features the message body only when working with SynchRequests

```
<?xml version="1.0"?>
<TestXml>This is a sample response message.</TestXml>
```

Error Codes and Message Catalog Entries

A response message may contain data relating to the processing of the request message, or it may contain error information if there were problems in fulfilling the request.

The status code describes the nature of the response message. The following table describes possible request message status codes and their meaning.

Value	Meaning	Description
0	Success	The message transport and processing were successful.
10	Retry	The transport was not successful. PeopleSoft Integration Broker will perform its retry logic and send the message again.
20	Error	An error occurred.
30	Duplicate message	The transaction ID for the message has already been received.
40	Acknowledgement error	This status is used for SOAP messages and indicates that the contents of the data is not proper, but the transport was successful.
50	Acknowledgement hold	Used for asynchronous chunking of messages from PeopleSoft to PeopleSoft nodes when sending multiple message segments.

All PeopleSoft Integration Broker error messages are stored in the message catalog. A short and long description for every error can be found there. Catalog entries are given a number, and this number is used in the response messages.

Here is a sample error message:

```
Message-ID: <32004392.1143500580241.JavaMail.KCOLLIN2@PLE-KCOLLIN2>
Date: Mon, 27 Mar 2006 15:03:00 -0800 (PST)
```

Copyright © 1988-2007, Oracle. All rights reserved.

```
Mime-Version: 1.0
Content-Type: multipart/related;
 boundary="----=_Part_4_9069393.1143500580221"
Content-ID: PeopleSoft-Integration-Broker-Internal-Mime-Message
PeopleSoft-ToolsRelease: 8.49

------=_Part_25_2235074.1008270392277
Content-Type: text/plain; charset=UTF-8
Content-Transfer-Encoding: 8bit
Content-Disposition: inline
Content-ID: IBInfo

<?xml version="1.0"?><IBInfo><Status><StatusCode>10</StatusCode><MsgSet>158</Msg⇒
Set>
<MsgID>10721</MsgID><Parameters count="1"><Parm>404</Parm></Parameters>
<DefaultTitle>Integration Gateway Error</DefaultTitle></Status></IBInfo>
------=_Part_25_2235074.1008270392277--
```

All PeopleSoft Integration Broker errors use message set *158*. The actual error seen here is *10721*. Going to the message catalog, the description for message set *158*, error *10721* is:

```
Message Text: Integration Gateway - External System Contact Error

Explanation: Integration Gateway was not able to contact the external system.
The network location specified may be incorrect, or the site is permanently
or temporarily down.
```

Therefore this error was created by the integration gateway when it tried to send a request message to an external system.

Local Compression

The integration engine compresses and base64–encodes messages destined for the PeopleSoft listening connector on its local integration gateway.

Asynchronous messages are always compressed and base64 encoded when sent to the integration gateway.

For synchronous messages, in PSADMIN you can set a threshold message size above which messages are compressed. The setting is shown here:

```
Values for config section - Integration Broker
    Min Message Size For Compression=10000

Do you want to change any values (y/n)? [n]:
```

The value is the message size in bytes; the default value is *10000* (10 kilobytes). You can specify a setting of *0* to compress all messages.

To turn off compression, set the value to *-1*.

Warning! Turning compression off can negatively impact system performance when transporting synchronous messages greater than 1 MB. As a result, you should turn off compression only during integration development and testing.

 Copyright © 1988-2007, Oracle. All rights reserved.

Note. This setting does not affect the compression of messages that the integration gateway sends using its target connectors.

See Also

Enterprise PeopleTools 8.49 PeopleBook: System and Server Administration, "Setting Application Server Domain Parameters"

Accessing IBInfo Elements Using PeopleCode

You can use the PeopleCode Message object to access IBRequest and IBResponse IBInfo data.

The following example demonstrates reading target connector information on a notification method for a rowset-based asynchronous message.

```
Local MESSAGE &MSG;
String &strReturn;

&MSG = %IntBroker.GetMessage();

For &i = 1 To &MSG.IBInfo.IBConnectorInfo.GetNumberOfConnectorProperties()

   &strReturn= &MSG.IBInfo.IBConnectorInfo.GetQueryStringArgName(&i);
   &strReturn= &MSG.IBInfo.IBConnectorInfo.GetQueryStringArgValue(&i);

End-For;
```

The following example demonstrates reading target connector information on notification method for a nonrowset-based asynchronous message.

```
method OnNotify
   /+ &_MSG as Message +/
   /+ Extends/implements PS_PT:Integration:INotificationHandler.OnNotify +/
   /* Variable Declaration */

   integer &i;

   string &&strReturn;

   xmldoc &xmldoc;

   For &i = 1 To &MSG.IBInfo.IBConnectorInfo.GetNumberOfConnectorProperties()

      &strReturn = &MSG.IBInfo.IBConnectorInfo.GetQueryStringArgName(&i);
      &strReturn = &MSG.IBInfo.IBConnectorInfo.GetQueryStringArgValue(&i);

   End-For;

   /* access the content data */

&xmldoc = &MSG.GetXmlDoc();
```

Copyright © 1988-2007, Oracle. All rights reserved.

```
end-method;
```

PeopleSoft Rowset-Based Message Format

This section discusses the PeopleSoft rowset-based message format and discusses:

- FieldTypes section of a rowset-based message.
- MsgData section of a rowset-based message.
- PeopleSoft rowset-based message example.
- PeopleSoft timestamp format.
- Schema restrictions.

This section also provides an example of a rowset-based message.

See Also

Chapter 8, "Understanding Supported Message Structures," Message Parts Structures, page 108

Understanding the PeopleSoft Rowset-Based Message Format

To work with rowset-based messages—the PeopleSoft native format—you define a message in the PeopleSoft Pure Internet Architecture, insert records into the message definition in a hierarchical structure, and then populate the message and manipulate its contents by using the PeopleCode Rowset and Message classes. Externally, the message is transmitted as XML with a prescribed PeopleSoft schema.

The PeopleSoft message schema includes a PSCAMA record, which PeopleTools adds to every level of the message structure to convey basic information about the message and its data rows.

The Rowset and IntBroker classes are recommended for messaging between PeopleSoft applications. If a message is populated with data from a PeopleSoft application's database or component buffer, the Message class is best for handling that data.

Record and Field Aliases

You can specify an alias for any record or field in a rowset-based message definition. Each node participating in a transaction maintains its own independent definition of the message and its versions, including record and field names and their aliases.

When you send a message with an alias defined and the message is converted to XML for sending, only the alias appears in the XML. If you don't specify an alias, the original name is used. If the service operation is routed to multiple target nodes with different record or field naming schemes, you create for each target a separate service operation version with its own aliases and send each version separately.

The only requirement for a successful transaction is that the record and field names in the XML match either the original names or the aliases that are defined for that message and version at the node receiving the message. This behavior applies to both request and synchronous response messages, but typically only the source node applies aliases to accommodate the target node's naming scheme in both directions.

 Copyright © 1988-2007, Oracle. All rights reserved.

In a synchronous transaction, the request and response messages can be completely different from each other. Upon receiving a synchronous request, the target node generates and sends a response message. Upon receiving the response, the source node uses its defined aliases to find and reapply its original record and field names. The resulting inbound message contains the original names that were defined at the source node, not the aliases. Therefore, both the sending and receiving PeopleCode at the source node should expect to work with the source node's original record and field names.

See Also

Chapter 8, "Understanding Supported Message Structures," PSCAMA, page 99

Chapter 12, "Sending and Receiving Messages," Understanding Integration PeopleCode, page 207

Chapter 20, "Applying Filtering, Transformation and Translation," page 373

Rowset-Based Message Template

The following template shows the overall structure of a message in the PeopleSoft rowset-based message format:

```
<?xml version="1.0"?>
   <psft_message_name>
      <FieldTypes>
         ...
      </FieldTypes>
      <MsgData>
         <Transaction>
            ...
         </Transaction>
      </MsgData>
   </psft_message_name>
```

Note. *Psft_message_name* is the name of the message definition in the PeopleSoft database. Integration Broker inserts this message content into a standard PeopleSoft XML message wrapper for transmission.

FieldTypes Section

Each PeopleSoft message includes *field type* information. Fieldtype information conveys the name of each data record and its constituent fields, along with each field's data type. Your receiving application can use this information to validate data types. The field type information is contained in the *FieldTypes* section of the message.

There are two FieldTypes tags:

- Each record tag consists of the name of a record, followed by a *class* attribute with a single valid value: *R*. The record tag encloses that record's field tags.
- Each field tag consists of the name of a field, followed by a *type* attribute with three valid values: *CHAR* for a character field, *DATE* for a date field, and *NUMBER* for a numeric field.

Following is a simple FieldTypes template.

```
<FieldTypes>
   <recordname1 class="R">
      <fieldname1 type="CHAR"/>
```

Copyright © 1988-2007, Oracle. All rights reserved.

```
      <fieldname2 type="DATE"/>
      <fieldname3 type="NUMBER"/>
   </recordname1>
   <recordname2 class="R">
      <fieldname4 type="NUMBER"/>
   </recordname2>
<FieldTypes>
```

Note. Third-party sending applications must include a valid FieldTypes section in each message. The PeopleSoft system expects fieldtype information for each record and field in the message.

MsgData Section

In addition to fieldtype information, each PeopleSoft message contains data content in the *MsgData* section of the message. Between the MsgData tags are one or more *Transaction* sections. Each transaction represents one row of data.

Between the Transaction tags is a rowset hierarchy of records and fields. The record tags at each level contain the fields for that record, followed by any records at the next lower level.

The last record within a transaction is a fully specified PeopleSoft Common Application Message Attributes (PSCAMA) record, which provides information about the entire transaction. Immediately following the closing tag of *every* record below level 0 is a PSCAMA record containing only the AUDIT_ACTN field that specifies the action for that record.

Simple MsgData Template

Following is a simple MsgData template.

Note. The PSCAMA PUBLISH_RULE_ID and MSGNODENAME fields (shown emphasized) are used internally by certain PeopleSoft utilities, but third-party systems can generally ignore them and don't need to include them in messages.

```
<MsgData>
   <Transaction>
      <level0recname1 class="R">
         <fieldname1>value</fieldname1>
         <fieldname2>value</fieldname2>
         <level1recname1 class="R">
            <fieldname3>value</fieldname3>
            <fieldname4>value</fieldname4>
         </level1recname1>
         <PSCAMA class="R">
            <AUDIT_ACTN>value</AUDIT_ACTN>
         </PSCAMA>
         <level1recname2 class="R">
            <fieldname5>value</fieldname5>
         </level1recname2>
         <PSCAMA class="R">
            <AUDIT_ACTN>value</AUDIT_ACTN>
         </PSCAMA>
      </level0recname1>
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
        <level0recname2 class="R">
           <fieldname6>value</fieldname6>
        </level0recname2>
        <PSCAMA class="R">
           <LANGUAGE_CD>value</LANGUAGE_CD>
           <AUDIT_ACTN>value</AUDIT_ACTN>
           <BASE_LANGUAGE_CD>value</BASE_LANGUAGE_CD>
           <MSG_SEQ_FLG>value</MSG_SEQ_FLG>
           <PROCESS_INSTANCE>value</PROCESS_INSTANCE>
          <PUBLISH_RULE_ID>value</PUBLISH_RULE_ID>
          <MSGNODENAME>value</MSGNODENAME>
        </PSCAMA>
      <Transaction>
</MsgData>
```

See Also

Chapter 8, "Understanding Supported Message Structures," PSCAMA, page 99

PSCAMA

PeopleTools adds the PSCAMA record to every level of the message structure during processing. It isn't accessible in the message definition, but you can reference it as part of the Message object in the sending and receiving PeopleCode, and you can see it in the Integration Broker Monitor. PeopleCode processes this record the same way as any other record.

Note. PSCAMA records are automatically included in messages only if you insert database records to define the message structure. You can use the PeopleCode XmlDoc class to handle an inbound message containing PSCAMA records, but the PeopleCode Message class is much better suited for this.

PSCAMA contains fields that are common to all messages. The <PSCAMA> tag repeats for each row in each level of the transaction section of the message. The sender can set PSCAMA fields to provide basic information about the message; for example, to indicate the message language or the type of transaction a row represents. When receiving a message, your PeopleCode should inspect the PSCAMA records for this information and respond accordingly.

PSCAMA Record Definition

The PSCAMA record definition includes the following fields:

LANGUAGE_CD Indicates the language in which the message is generated, so the receiving application can take that information into account when processing the message. When sending a message with component PeopleCode, the system sets this field to the user's default language code.

AUDIT_ACTN Identifies each row of data as an Add, Change, or Delete action.

BASE_LANGUAGE_CD (Optional.) Indicates the base language of the sending database. This is used by generic, full-table subscription PeopleCode to help determine which tables to update.

MSG_SEQ_FLG (Optional.) Supports the transmission of large transactions that may span multiple messages. Indicates whether the message is a header (*H*) or trailer (*T*) or contains data (blank). The header and trailer messages don't contain

Copyright © 1988-2007, Oracle. All rights reserved.

message data. The receiving system can use this information to determine the start and end of the set of messages and initiate processes accordingly. For example, the header message might cause staging tables to be cleared, while the trailer might indicate that all of the data has been received and an update job should be initiated.

PROCESS_INSTANCE (Optional.) Process instance of the batch job that created the message. Along with the sending node and publication ID, the receiving node can use this to identify a group of messages from the sending node.

PUBLISH_RULE_ID (Optional.) Indicates the publish rule that is invoked to create the message. This is used by routing PeopleCode to locate the appropriate chunking rule, which then determines to which nodes the message should be sent. Third-party applications can ignore this field.

MSGNODENAME (Optional.) The node to which the message should be sent. This field is passed to the Publish utility by the Application Engine program. Routing PeopleCode must look for a value in this field and return that value to the application server. Third-party applications can ignore this field.

Language Codes

Each message can contain only one language code (the LANGUAGE_CD field) in the first PSCAMA record.

PeopleSoft language codes contain three characters and are mapped to corresponding International Organization for Standardization (ISO) locale codes in an external properties file. This mapping enables the PeopleSoft Pure Internet Architecture to derive certain defaults from the ISO locales that are stored in a user's browser settings. Your PeopleSoft application is delivered with a set of predefined language codes; you can define your own codes, as well.

Note. There can be only one language code for the entire message. To send messages in multiple languages, send multiple messages.

See *Enterprise PeopleTools 8.49 PeopleBook: Global Technology*, "Controlling International Preferences".

Audit Action Codes

A PSCAMA record appears following each row of the message. At a minimum, it contains an audit action code (the AUDIT_ACTN field), denoting the action to be applied to the data row. The audit action is required so that the receiving system knows how to process the incoming data.

The valid audit action codes match those that are used in the PeopleSoft audit trail processing: *A, C, D, K, N, O,* and blank.

The audit action values are set by the system or by the sending PeopleCode to specify that a record should be added, changed, or deleted:

 Copyright © 1988-2007, Oracle. All rights reserved.

Audit Action Code	Description
A	Add a noneffective or effective-dated row. To add an effective-dated row, the value is *A*. If you populate the row data by using the CopyRowsetDeltaOriginal method in the PeopleCode Message class, an additional record is created with an audit action value of *O,* containing the original values of the current effective-dated row.
C	Change non-key values in a row.
D	Delete a row
K	If you change at least one key value in a row (in addition to any non-key values) and then populate the row data by using the CopyRowsetDeltaOriginal or CopyRowsetDelta methods in the Message class, an additional record is created with an audit action value of *K,* containing the original values of the current effective-dated row.
N	Change at least one key value in a row (in addition to any non-key values).
O	If you change non-key values in a row and populate the row data by using the CopyRowsetDeltaOriginal method in the Message class, an additional record is created with an audit action value of *O,* containing the original values of the current effective-dated row.
Blank	Default value. If a row's content hasn't changed, the value is blank. This audit action code is also used to tag the parents of rows that have changed.

Other PSCAMA Considerations

You can update values on the PSCAMA record just like any other record in the message definition before sending the message.

The receiving process can access the fields in this record just like any other fields in the message.

The size of the PSCAMA record varies. In particular, notice a difference between the first PSCAMA record and the ones that follow. The first record contains all of the fields, while the other PSCAMA records have only the AUDIT_ACTN field, which is the only field that can differ for each row of data.

Although the first PSCAMA record is always present, not all of the remaining PSCAMA records are sent in the message. If a <PSCAMA> tag is not included for a specific row, you can assume that the row hasn't changed. In addition, if the <AUDIT_ACTN> tag is blank or null, you can also assume the row hasn't changed.

If you're receiving a message that has incremental changes, only the rows that have changed *and their parent rows* are present in the message.

You can view an example of an outbound message with the PSCAMA records inserted by testing your message definition using the Schema Tester.

See *Enterprise PeopleTools 8.49 PeopleBook: PeopleSoft Integration Testing Utilities and Tools*, "Using the Schema Tester Utility".

Copyright © 1988-2007, Oracle. All rights reserved.

Identifying Changes to Field-Level Attributes

When sending and receiving messages, all blank data values get stripped. As a result, you cannot determine if a field value is blank by definition, or if its value was stripped in the messaging process.

The PeopleCode CopyRowset functions CopyRowset, CopyRowsetDelta and CopyRowsetDeltaOriginal, feature an IsChanged attribute that automatically gets set to identify fields that have been changed. Any field that has been changed displays the attribute *IsChanged="Y"*.

Note. The IsChanged attribute applies only to rowset-based messages.

It does not apply to nonrowset-based messages, including container messages, or part messages.

For example:

```
<QE_ACNUMBER IsChanged="Y">2</QE_ACNUMBER>
```

Fields that had data and then were blanked contain the IsChanged attribute.

For example:

```
<DESCRLONG IsChanged="Y"/>
```

Fields that were always blank and thus were not changed do not feature this attribute. For example:

```
<QE_NAVDESC/>
```

If you are writing subscription PeopleCode you reference the IsChanged value of the field in the message rowset, as always. However, the blanks appear with the attribute *IsChanged="Y"*.

PeopleSoft Timestamp Format

The PeopleSoft format for all timestamps is ISO-8601. If any message fields are type timestamp, the following format is used:

```
CCYY-MM-DDTHH:MM:SS.ssssss+/-hhmm
```

Note. The ISO format specifies that the `+/-hhmm` parameter is optional, but PeopleSoft requires it. All date and time stamps in the header and the body of the message must include the Greenwich Mean Time (GMT) offset as `+/-hhmm`. This ensures that the timestamp is correctly understood by the receiving application.

Schema Restrictions

For stronger schema validation control, certain restrictions apply to fields having the following formats:

- Mixed case
- Name.
- Phone number
- Social security number.
- Uppercase.
- Zip code.

Note. These restrictions apply to rowset-based messages and rowset-based message parts.

The restrictions for each are shown in the following example:

 Copyright © 1988-2007, Oracle. All rights reserved.

```
<xsd:simpleType name="BASE_LANGUAGE_CD_TypeDef">
    <xsd:annotation>
      <xsd:documentation>BASE_LANGUAGE_CD is a character of length 3.
        Allows Uppercase characters including numbers
      </xsd:documentation>
    </xsd:annotation>
    <xsd:restriction base="xsd:string">
      <xsd:maxLength value="3"/>
      <xsd:whiteSpace value="preserve"/>
      <xsd:pattern value="([A-Z]|[0-9]|\p{Z}|\p{P}|\p{Lu})*"/>
    </xsd:restriction>
  </xsd:simpleType>
```

Rowset-Based Message Example

The message data is enclosed in a tag with the name of the message, and consists of one FieldTypes section followed by one MsgData section. The FieldTypes section describes the records and fields that appear in the MsgData section, which contains the actual data.

Note. The PSCAMA record requires fieldtype information just like any other record.

```
<SDK_BUS_EXP_APPR_MSG>
   <FieldTypes>
      <SDK_BUS_EXP_PER class="R">
         <SDK_EMPLID type="CHAR"/>
         <SDK_EXP_PER_DT type="DATE"/>
         <SDK_SUBMIT_FLG type="CHAR"/>
         <SDK_INTL_FLG type="CHAR"/>
         <SDK_APPR_STATUS type="CHAR"/>
         <SDK_APPR_INSTANCE type="NUMBER"/>
         <SDK_DESCR type="CHAR"/>
         <SDK_COMMENTS type="CHAR"/>
      </SDK_BUS_EXP_PER>
      <SDK_DERIVED class="R">
         <SDK_BUS_EXP_SUM type="NUMBER"/>
      </SDK_DERIVED>
      <SDK_BUS_EXP_DTL class="R">
         <SDK_CHARGE_DT type="DATE"/>
         <SDK_EXPENSE_CD type="CHAR"/>
         <SDK_EXPENSE_AMT type="NUMBER"/>
         <SDK_CURRENCY_CD type="CHAR"/>
         <SDK_BUS_PURPOSE type="CHAR"/>
         <SDK_DEPTID type="CHAR"/>
      </SDK_BUS_EXP_DTL>
      <PSCAMA class="R">
         <LANGUAGE_CD type="CHAR"/>
         <AUDIT_ACTN type="CHAR"/>
         <BASE_LANGUAGE_CD type="CHAR"/>
         <MSG_SEQ_FLG type="CHAR"/>
         <PROCESS_INSTANCE type="NUMBER"/>
```

Copyright © 1988-2007, Oracle. All rights reserved.

```
   </PSCAMA>
</FieldTypes>
<MsgData>
   <Transaction>
      <SDK_BUS_EXP_PER class="R">
         <SDK_EMPLID>8001</SDK_EMPLID>
         <SDK_EXP_PER_DT>1998-08-22</SDK_EXP_PER_DT>
         <SDK_SUBMIT_FLG>N</SDK_SUBMIT_FLG>
         <SDK_INTL_FLG>N</SDK_INTL_FLG>
         <SDK_APPR_STATUS>P</SDK_APPR_STATUS>
         <SDK_APPR_INSTANCE>0</SDK_APPR_INSTANCE>
         <SDK_DESCR>Regional Users Group Meeting</SDK_DESCR>
         <SDK_COMMENTS>Attending Northeast Regional Users Group
         Meeting and presented new release functionality.
         </SDK_COMMENTS>
         <SDK_BUS_EXP_DTL class="R">
            <SDK_CHARGE_DT>1998-08-22</SDK_CHARGE_DT>
            <SDK_EXPENSE_CD>10</SDK_EXPENSE_CD>
            <SDK_EXPENSE_AMT>45.690</SDK_EXPENSE_AMT>
            <SDK_CURRENCY_CD>USD</SDK_CURRENCY_CD>
            <SDK_BUS_PURPOSE>Drive to Meeting</SDK_BUS_PURPOSE>
            <SDK_DEPTID>10100</SDK_DEPTID>
         </SDK_BUS_EXP_DTL>
         <PSCAMA class="R">
            <AUDIT_ACTN>A</AUDIT_ACTN>
         </PSCAMA>
         <SDK_BUS_EXP_DTL class="R">
            <SDK_CHARGE_DT>1998-08-22</SDK_CHARGE_DT>
            <SDK_EXPENSE_CD>09</SDK_EXPENSE_CD>
            <SDK_EXPENSE_AMT>12.440</SDK_EXPENSE_AMT>
            <SDK_CURRENCY_CD>USD</SDK_CURRENCY_CD>
            <SDK_BUS_PURPOSE>City Parking</SDK_BUS_PURPOSE>
            <SDK_DEPTID>10100</SDK_DEPTID>
         </SDK_BUS_EXP_DTL>
         <PSCAMA class="R">
            <AUDIT_ACTN>A</AUDIT_ACTN>
         </PSCAMA>
      </SDK_BUS_EXP_PER>
      <SDK_DERIVED class="R">
         <SDK_BUS_EXP_SUM>58.13</SDK_BUS_EXP_SUM>
      </SDK_DERIVED>
      <PSCAMA class="R">
         <LANGUAGE_CD>ENG</LANGUAGE_CD>
         <AUDIT_ACTN>A</AUDIT_ACTN>
         <BASE_LANGUAGE_CD>ENG</BASE_LANGUAGE_CD>
         <MSG_SEQ_FLG></MSG_SEQ_FLG>
         <PROCESS_INSTANCE>0</PROCESS_INSTANCE>
      </PSCAMA>
   </Transaction>
</MsgData>
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
</SDK_BUS_EXP_APPR_MSG>
```

Nonrowset-Based Message Structures

This section discusses nonrowset-based message structures that you an use with PeopleSoft Integration Broker. This section discusses:

- XML DOC-compliant messages.
- SOAP-compliant messages.
- Non-XML files.

XML DOC-Compliant Messages

The World Wide Web Consortium (W3C) has established a DOM for accessing and manipulating structured data. A DOM specifies how data should be presented for access by programming languages, regardless of the data's actual internal representation. The DOM specifies a standardized application programming interface (API) that provides a consistent, familiar way to work with almost any data. This API—the XML DOM—enables you to create, retrieve, navigate, and modify messages.

You define an XML DOC-compliant message in the PeopleSoft Pure Internet Architecture by either uploading an XML file or entering an XML schema definition. The following example shows an XML DOM-compliant message schema:

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" targetNamespace=
"http://xmlns.oracle.com/Common/schemas/COMPANY" xmlns="http://xmlns.
oracle.com/Common/schemas/COMPANY" elementFormDefault="qualified">
  <xsd:element name="Company" type="CompanyType"/>
  <xsd:complexType name="CompanyType">
     <xsd:sequence>
        <xsd:element name="Person" type="PersonType"/>
        <xsd:element name="Product" type="ProductType"/>
     </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="PersonType">
     <xsd:sequence>
        <xsd:element name="Name" type="xsd:string"/>
        <xsd:element name="SSN" type="xsd:string"/>
     </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="ProductType">
     <xsd:sequence>
        <xsd:element name="Type" type="xsd:string"/>
     </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

Copyright © 1988-2007, Oracle. All rights reserved.

Then populate the message and manipulate its contents by using the PeopleCode XmlDoc class and built-in functions, which comply with the XML DOM. The XmlDoc class is well-suited for handling messages that are populated with XML data from an external file or uniform resources locator (URL).

Note. You can use the XmlDoc class to access inbound, rowset-based messages; however, the PeopleCode Message and Rowset classes handle the PeopleSoft native format more easily.

Use the XmlDoc class if *any* of the following is true:

- The message structure doesn't fit the PeopleSoft rowset model.
- The message data doesn't come from PeopleSoft database records.
- The third-party source or target node requires non-XML message data.

Although you can use the XmlDoc class to generate or process messages that use the SOAP protocol, the PeopleCode SoapDoc class is more efficient and is strongly recommended.

Note. Both SOAP and non-XML message data must be embedded in an XML wrapper, which you send and receive by using the XmlDoc class.

SOAP-Compliant Messages

The W3C SOAP specification defines synchronous transactions in a distributed web environment. SOAP is appropriate for Universal Description, Discovery, and Integration (UDDI) interactions, or to interact with SOAP-compliant servers.

You define a message in PeopleSoft Application Designer without inserting any records to define its structure, then populate the message and manipulate its contents by using the PeopleCode SoapDoc class and built-in functions, which comply with the W3C SOAP specification. The SoapDoc class is well-suited for messages that are populated with SOAP-compliant XML data.

SoapDoc complies with the W3C XML DOM specification. The SoapDoc class is based on the PeopleCode XmlDoc class, with some identical methods and properties. To send and receive SoapDoc messages, you must convert them to XmlDoc objects and use the XMLDoc built-in functions, SyncRequestXmlDoc and GetMessageXmlDoc. SoapDoc provides a property for handling the conversion easily.

Use the SoapDoc class if *all* of the following are true:

- The third-party source or target node requires SOAP-compliant messages.
- The third-party source or target node requires synchronous transactions.
- The message conforms to the SOAP specification.

See Also

Chapter 12, "Sending and Receiving Messages," Generating and Sending Messages, page 221

Chapter 12, "Sending and Receiving Messages," Receiving and Processing Messages, page 232

Non-XML Files

To send non-XML files through PeopleSoft Integration Broker to their destination, you must wrap them in the PeopleSoft non-XML message element, CDATA. However, when you send messages to third-party systems, the recipient systems may not be able to interpret that element.

 Copyright © 1988-2007, Oracle. All rights reserved.

If you are using the Publish or SyncRequest methods to send data, you can use the built-in function SetXMLDoc to remove the tags upon exiting the integration gateway or write a transformation to do so. If you choose neither of these options, the data remains in the wrapper through to the destination.

The following code example shows a non-XML file wrapped in the PeopleSoft non-XML message element for transport through PeopleSoft Integration Broker:

```
<?xml version="1.0"?>
<AsyncRequest>
  <data PsNonXml="Yes">
    <![CDATA[<?xml version="1.0"?>101 123456789 12345678
902
    0510145 60094101First Bank First Bank 5200 University 000001
    PPDDIRECT PAY020510020510000112345678000000162200000111 222
    0000001000USA0000001 USA0000001 000000111000000001627123456
    789131415511 0000001000 University 0123456780000
    002 820000000200123457890000000010000000000001000 123456780000001
    900000100000100000002001234578900000000100000000000100099999999999
    999999999999999999999999999999999999999999999999999999999999999999
    999999999999999999999999999999999999999999999999999999999999999999
    999999999999999999999999999999999999999999999999999999999999999999
    999999999999999999999999999999999999999999999999999999999999999999
    999999999999999999999999999999999999999999999999999999999999999999
    99999999999999999999999999999999999
   ]]>
    </data>
</AsyncRequest>
```

The following example shows an alternative way to wrap a non-XML file in the PeopleSoft non-XML message element for transport through PeopleSoft Integration Broker:

```
<?xml version="1.0"?>
<AsyncRequest psnonxml = 'Yes'>
  // or psnonxml can be mixed case
<AsyncRequest PsNonXml = 'Yes'>
    <![CDATA[<?xml version="1.0"?>101 123456789 12345678
902
    0510145 60094101First Bank First Bank 5200 University 000001
    PPDDIRECT PAY020510020510000112345678000000162200000111 222
    0000001000USA0000001 USA0000001 000000111000000001627123456
    789131415511 0000001000 University 0123456780000
    002 820000000200123457890000000010000000000001000 123456780000001
    900000100000100000002001234578900000000100000000000100099999999999999999
    999999999999999999999999999999999999999999999999999999999999999999999999
    999999999999999999999999999999999999999999999999999999999999999999999999
    999999999999999999999999999999999999999999999999999999999999999999999999
    999999999999999999999999999999999999999999999999999999999999999999999999
    999999999999999999999999999999999999999999999999999999999999999999999999
    ]]>
</AsyncRequest>
```

Copyright © 1988-2007, Oracle. All rights reserved.

See Also

Chapter 9, “Using Listening Connectors and Target Connectors,” Complying With Message Formatting and Transmission Requirements, page 129

Using Nonrowset-Based Messages in Service Operations Exposed as WSDL

If a nonrowset-based message is used in a service operation which will be exposed as a WSDL to third parties, the schema cannot be empty. The schema has to have at least the header elements, as shown in the following example:

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"/>
```

Message Parts Structures

This section discusses:

- Rowset-based message parts.
- Nonrowset-based message parts.

Understanding Message Part Structures

Message parts are rowset-based messages or nonrowset-based messages that you designate as a part message on the message definition. Message parts are used in container messages

Message parts can be re-used in multiple containers.

All parts in a container must be of the same type (Rowset-based or Nonrowset-based).

You create messages using the Message Builder page in the PeopleSoft Pure Internet Architecture.

See Also

Chapter 8, “Understanding Supported Message Structures,” PeopleSoft Rowset-Based Message Format, page 96

Chapter 8, “Understanding Supported Message Structures,” Nonrowset-Based Message Structures, page 105

Chapter 10, “Managing Messages,” page 177

Rowset-Based Message Parts

Rowset-based message parts provide all the ease of use of using rowsets, yet the generated XML message is industry standard and not PeopleSoft proprietary. Rowset-based message parts, like nonrowset-based parts, can only be part of a container type message.

These are the benefits of using Rowset-based parts:

 Copyright © 1988-2007, Oracle. All rights reserved.

- The XML schema generated is standard XML and not the PeopleSoft message format. Rowset-based message parts do not have a PSCAMA section, FieldTypes section, IsChanged attributes, and so forth.
- The message API for rowset-based parts is simple to use and understand.
- XML serialization and deserialization to and from part rowset is provided by Integration Broker framework
- You can use a CopyRowSet type method to populate the rowset from another rowset (component rowset).

The following example shows a sample schema from a rowset-based message part:

```
<?xml version="1.0"?>
<xsd:schema elementFormDefault="qualified" targetNamespace="http://xmlns.
oracle.com/Enterprise/Tools/schemas/Part_1.V1" xmlns="http://xmlns.oracle.
com/Enterprise/Tools/schemas/Part_1.V1" xmlns:xsd="http://www.w3.org/
2001/XMLSchema">
  <xsd:element name="Part_1" type="Part_1_TypeShape"/>
  <xsd:complexType name="Part_1_TypeShape">
    <xsd:sequence>
      <xsd:element name="First_Part" type="First_PartMsgDataRecord_TypeShape"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="First_PartMsgDataRecord_TypeShape">
    <xsd:sequence>
      <xsd:element name="QE_ACNUMBER" type="QE_ACNUMBER_TypeDef"/>
      <xsd:element name="QE_WAYPOINT_NBR" type="QE_WAYPOINT_NBR_TypeDef"/>
      <xsd:element minOccurs="0" name="QE_BEARING" type="QE_BEARING_TypeDef"/>
      <xsd:element minOccurs="0" name="QE_RANGE" type="QE_RANGE_TypeDef"/>
      <xsd:element minOccurs="0" name="QE_ALTITUDE" type="QE_ALTITUDE_TypeDef"/>
      <xsd:element minOccurs="0" name="QE_LATITUDE" type="QE_LATITUDE_TypeDef"/>
      <xsd:element minOccurs="0" name="QE_LONGITUDE" type="QE_LONGITUDE_TypeDef"/>
      <xsd:element name="QE_HEADING" type="QE_HEADING_TypeDef"/>
      <xsd:element name="QE_VELOCITIES" type="QE_VELOCITIES_TypeDef"/>
      <xsd:element minOccurs="0" name="QE_NAVDESC" type="QE_NAVDESC_TypeDef"/>
    </xsd:sequence>
    <xsd:attribute fixed="R" name="class" type="xsd:string" use="required"/>
  </xsd:complexType>
  <xsd:simpleType name="QE_ACNUMBER_TypeDef">
    <xsd:annotation>
      <xsd:documentation>QE_ACNUMBER is a number of length 10 with a decimal
        position of 0</xsd:documentation>
    </xsd:annotation>
    <xsd:restriction base="xsd:integer">
      <xsd:totalDigits value="10"/>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="QE_WAYPOINT_NBR_TypeDef">
    <xsd:annotation>
      <xsd:documentation>QE_WAYPOINT_NBR is a number of length 3 with a decimal
        position of 0</xsd:documentation>
    </xsd:annotation>
    <xsd:restriction base="xsd:integer">
```

Copyright © 1988-2007, Oracle. All rights reserved.

```
      <xsd:totalDigits value="3"/>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="QE_BEARING_TypeDef">
    <xsd:annotation>
      <xsd:documentation>QE_BEARING is a character of length 10</xsd:⇒
documentation>
    </xsd:annotation>
    <xsd:restriction base="xsd:string">
      <xsd:maxLength value="10"/>
      <xsd:whiteSpace value="preserve"/>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="QE_RANGE_TypeDef">
    <xsd:annotation>
      <xsd:documentation>QE_RANGE is a character of length 10</xsd:documentation>
    </xsd:annotation>
    <xsd:restriction base="xsd:string">
      <xsd:maxLength value="10"/>
      <xsd:whiteSpace value="preserve"/>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="QE_ALTITUDE_TypeDef">
    <xsd:annotation>
      <xsd:documentation>QE_ALTITUDE is a character of length 10</xsd:⇒
documentation>
    </xsd:annotation>
    <xsd:restriction base="xsd:string">
      <xsd:maxLength value="10"/>
      <xsd:whiteSpace value="preserve"/>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="QE_LATITUDE_TypeDef">
    <xsd:annotation>
      <xsd:documentation>QE_LATITUDE is a character of length 15
      </xsd:documentation>
    </xsd:annotation>
    <xsd:restriction base="xsd:string">
      <xsd:maxLength value="15"/>
      <xsd:whiteSpace value="preserve"/>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="QE_LONGITUDE_TypeDef">
    <xsd:annotation>
      <xsd:documentation>QE_LONGITUDE is a character of length 15
      </xsd:documentation>
    </xsd:annotation>
    <xsd:restriction base="xsd:string">
      <xsd:maxLength value="15"/>
      <xsd:whiteSpace value="preserve"/>
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="QE_HEADING_TypeDef">
    <xsd:annotation>
      <xsd:documentation>QE_HEADING is a character of length 4
     </xsd:documentation>
    </xsd:annotation>
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="MAG"/>
      <xsd:enumeration value="TRUE"/>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="QE_VELOCITIES_TypeDef">
    <xsd:annotation>
      <xsd:documentation>QE_VELOCITIES is a character of length 4
      </xsd:documentation>
    </xsd:annotation>
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="ADC"/>
      <xsd:enumeration value="GPS"/>
      <xsd:enumeration value="INS"/>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="QE_NAVDESC_TypeDef">
    <xsd:annotation>
      <xsd:documentation>QE_NAVDESC is a character of length 30
      </xsd:documentation>
    </xsd:annotation>
    <xsd:restriction base="xsd:string">
      <xsd:maxLength value="30"/>
      <xsd:whiteSpace value="preserve"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:schema>
```

Nonrowset-Based Message Parts

A nonrowset-based message part schema is similar to a regular nonrowset-based message, however a nonrowset-based except a nonrowset-based message part can be reused in multiple containers.

Message Container Structures

Message container structures hold rowset-based or nonrowset-based message part structures. All message parts assigned to a container must of the same type, rowset-based or nonrowset-based.

A message container is always a nonrowset-based message.

Copyright © 1988-2007, Oracle. All rights reserved.

You create container messages using the Message Builder in the PeopleSoft Pure Internet Architecture.

See Also

Chapter 8, "Understanding Supported Message Structures," Nonrowset-Based Message Structures, page 105

Chapter 10, "Managing Messages," page 177

Example 1: XML Schema of a Container Message with Rowset-Based Message Parts

The following example shows a sample schema of a container message with three rowset-based message parts:

```
<?xml version="1.0"?>
<xsd:schema elementFormDefault="unqualified" targetNamespace="http://xmlns.
oracle.com/Enterprise/Tools/schemas/Part_Container.V1"
xmlns="http://xmlns.oracle.com/Enterprise/Tools/schemas/Part_Container.V1"
xmlns:Part_1.V1="http://xmlns.oracle.com/Enterprise/Tools/schemas/Part_1.V1"
xmlns:Part_2.V1="http://xmlns.oracle.com/Enterprise/Tools/schemas/Part_2.V1"
xmlns:Part_3.V1="http://xmlns.oracle.com/Enterprise/Tools/schemas/Part_3.V1"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:import namespace="http://xmlns.oracle.com/Enterprise/Tools/schemas/
    Part_1.V1" schemaLocation="http://kcollin2042803:5000/PSIGW/PeopleSoft
     ServiceListeningConnector?Operation=GetSchema&amp;xsd=Part_1.V1"/>
  <xsd:import namespace="http://xmlns.oracle.com/Enterprise/Tools/schemas/
     Part_3.V1" schemaLocation="http://kcollin2042803:5000/PSIGW/PeopleSoft
     ServiceListeningConnector?Operation=GetSchema&amp;xsd=Part_3.V1"/>
  <xsd:import namespace="http://xmlns.oracle.com/Enterprise/Tools/schemas/
     Part_2.V1" schemaLocation="http://kcollin2042803:5000/PSIGW/PeopleSoft
     ServiceListeningConnector?Operation=GetSchema&amp;xsd=Part_2.V1"/>
  <xsd:element name="Part_Container" type="Part_ContainerType"/>
  <xsd:complexType name="Part_ContainerType">
    <xsd:sequence>
      <xsd:element maxOccurs="unbounded" minOccurs="0" name="Part_1" type="
         Part_1.V1:Part_1_TypeShape"/>
      <xsd:element maxOccurs="10" minOccurs="0" name="Part_3" type="Part_3.V1:
         Part_3_TypeShape"/>
      <xsd:element maxOccurs="unbounded" minOccurs="0" name="Part_2" type="
         Part_2.V1:Part_2_TypeShape"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

Example 2: XML Schema of a Container Message with Nonrowset-Based Message Parts

The following example shows a sample schema from a container message that contains three nonrowset-based parts:

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

 Copyright © 1988-2007, Oracle. All rights reserved.

```
<xsd:schema elementFormDefault="unqualified" targetNamespace="http://xmlns.
 oracle.com/Enterprise/Tools/schemas/NonRowSetContainer.v1"
xmlns="http://xmlns.oracle.com/Enterprise/Tools/schemas/NonRowSetContainer.v1"
xmlns:Part_One_NonRowset.v1="http://xmlns.oracle.com/Enterprise/Tools/
   schemas/Part_One.v1"
xmlns:Part_Three_NonRowset.v1="http://xmlns.oracle.com/Enterprise/Tools/
   schemas/Part_One.v1"
xmlns:Part_Two_NonRowset.v1="http://xmlns.oracle.com/Enterprise/Tools/
   schemas/Part_One.v1"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:import schemaLocation="http://kcollin2042803:5000/PSIGW/PeopleSoft
    ServiceListeningConnector?Operation=GetSchema&amp;xsd=Part_One_NonRowset.v1"/>
  <xsd:import schemaLocation="http://kcollin2042803:5000/PSIGW/PeopleSoft
     ServiceListeningConnector?Operation=GetSchema&amp;xsd=Part_Two_NonRowset.v1"⇒
/>
  <xsd:import schemaLocation="http://kcollin2042803:5000/PSIGW/PeopleSoft
     ServiceListening Connector?Operation=GetSchema&amp;xsd=Part_Three_Non⇒
Rowset.v1"/>
  <xsd:element name="NonRowSetContainer" type="NonRowSetContainerType"/>
  <xsd:complexType name="NonRowSetContainerType">
    <xsd:sequence>
      <xsd:element maxOccurs="unbounded" minOccurs="0" name="Part_One_NonRowset"
        type="Part_One_NonRowset.v1:Part_One_TypeShape"/>
      <xsd:element maxOccurs="unbounded" minOccurs="0" name="Part_Two_NonRowset"
        type="Part_Two_NonRowset.v1:Part_One_TypeShape"/>
      <xsd:element maxOccurs="unbounded" minOccurs="0" name="Part_Three_NonRowset"
         type="Part_Three_NonRowset.v1:Part_One_TypeShape"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

Copyright © 1988-2007, Oracle. All rights reserved.

 Copyright © 1988-2007, Oracle. All rights reserved.

CHAPTER 9

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.

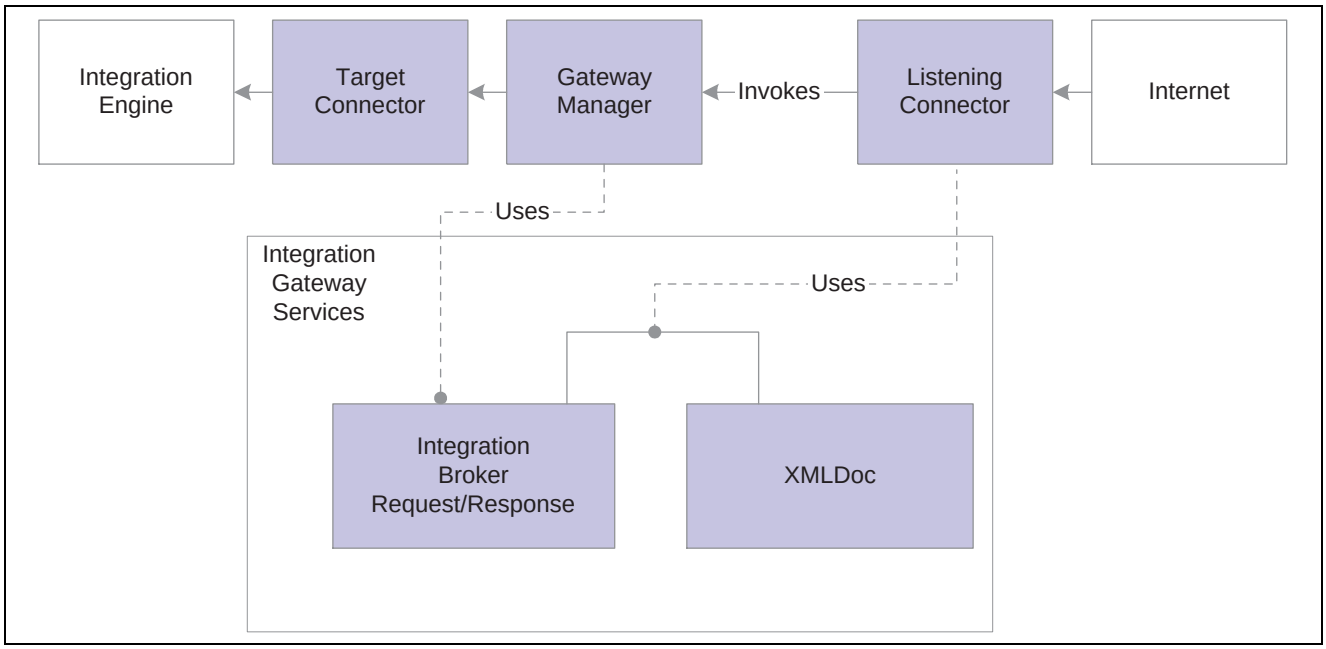
Understanding Listening Connectors and Target Connectors

This section discusses listening connectors, target connectors and target connector properties.

Listening Connectors

Listening connectors receive message 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 message from an external system into the integration engine through a listening connector:

Copyright © 1988-2007, Oracle. All rights reserved.



Message 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. For all connectors in the following table except the service listening connector, the gateway provides a matching target 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 9, "Using Listening Connectors and Target Connectors," Working With the PeopleSoft Connectors, page 122.
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 9, "Using Listening Connectors and Target Connectors," Working With the HTTP Connectors, page 123.

Copyright © 1988-2007, Oracle. All rights reserved.

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 9, "Using Listening Connectors and Target Connectors," Working With the PeopleSoft 8.1 Connectors, page 139.
JMS listening connector	This connector enables JMS provider systems to communicate with the gateway using standard JMS protocols. See Chapter 9, "Using Listening Connectors and Target Connectors," Working With the JMS Connectors, page 141.
AS2 listening connector	The AS2 listening connector enables you to receive request messages in AS2 format. See Chapter 9, "Using Listening Connectors and Target Connectors," Working With the AS2 Connectors, page 164.
PeopleSoft service 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.

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

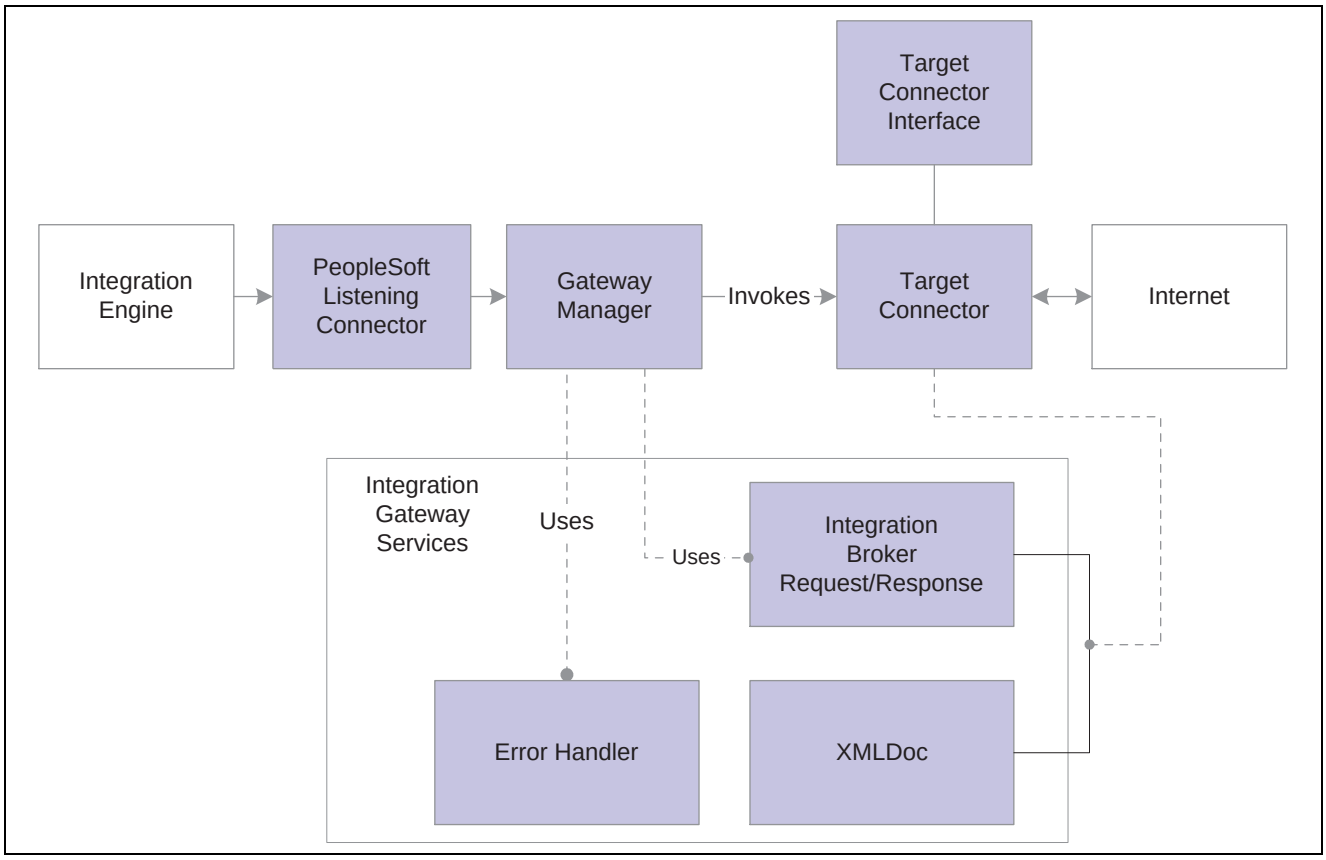
Null Characters in Messages

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.

Target Connectors

Target connectors generate message 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:

Copyright © 1988-2007, Oracle. All rights reserved.



Message 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:

Copyright © 1988-2007, Oracle. All rights reserved.

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 BEA Jolt connection in the PeopleSoft internal messaging format. Use this connector to send messages only to PeopleSoft applications that use PeopleSoft Integration Broker. **Note.** BEA Jolt is a Java-based interface that extends BEA 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 9, "Using Listening Connectors and Target Connectors," Working With the PeopleSoft Connectors, page 122.
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 9, "Using Listening Connectors and Target Connectors," Working With the HTTP Connectors, page 123.
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 9, "Using Listening Connectors and Target Connectors," Working With the PeopleSoft 8.1 Connectors, page 139.
JMS target connector	This connector enables the gateway to communicate with JMS provider systems using standard JMS protocols. See Chapter 9, "Using Listening Connectors and Target Connectors," Working With the JMS Connectors, page 141.
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 9, "Using Listening Connectors and Target Connectors," Working With the FTP Target Connector, page 157.
AS2 target connector	The AS2 target connector enables you to send messages in AS2 format. See Chapter 9, "Using Listening Connectors and Target Connectors," Working With the AS2 Connectors, page 164.

Copyright © 1988-2007, Oracle. All rights reserved.

Connector	Description
SMTP target connector	With this connector, the gateway can send messages to an SMTP server using the PUT command. See Chapter 9, "Using Listening Connectors and Target Connectors," Working With the SMTP Target Connector, page 174.
Simple file target connector	With this connector, the gateway saves outbound messages as XML files. See Chapter 9, "Using Listening Connectors and Target Connectors," Working With the Simple File Target Connector, page 156.
GetMail target connector	This connector provides functionality specific to the PeopleSoft Multichannel Framework. See *Enterprise PeopleTools 8.49 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 7, "Managing Integration Gateways," Administering Integration Gateways, page 54.

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 19, "Adding and Configuring Nodes," page 361.

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 Chapter 18, "Managing Routing Definitions," Overriding Gateway and Connector Properties, page 346.

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

See Chapter 12, "Sending and Receiving Messages," Setting and Overriding Target Connector Properties at Runtime, page 228.

 Copyright © 1988-2007, Oracle. All rights reserved.

Passwords

You must encrypt all required and optional target connector passwords.

See Chapter 7, "Managing Integration Gateways," Encrypting Passwords, page 66.

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 19, "Adding and Configuring Nodes," page 361.

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.

Copyright © 1988-2007, Oracle. All rights reserved.

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 BEA 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 BEA Jolt connect string properties in the integrationGateway.properties file to determine where to send the messages.

See Chapter 7, "Managing Integration Gateways," Setting General Connection Properties, page 69.

 Copyright © 1988-2007, Oracle. All rights reserved.

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
Operation	Specify the external alias name.

Copyright © 1988-2007, Oracle. All rights reserved.

Parameter	Description
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. **Note.** 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. **Note.** 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.
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 7, "Managing Integration Gateways," Setting General Connection Properties, page 69.

 Copyright © 1988-2007, Oracle. All rights reserved.

Parameter	Description
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 Chapter 11, "Managing Service Operation Queues," Applying Queue Partitioning, page 199.
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 9, "Using Listening Connectors and Target Connectors," Complying With Message Formatting and Transmission Requirements, page 129.

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.

Copyright © 1988-2007, Oracle. All rights reserved.

- 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 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*.

 Copyright © 1988-2007, Oracle. All rights reserved.

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 9, "Using Listening Connectors and Target Connectors," Running Integration Gateways Behind Proxy Servers, page 137.

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*. (Default.) Outbound messages are wrapped in SOAP format. • *N*. Outbound message are not wrapped in SOAP format.

Copyright © 1988-2007, Oracle. All rights reserved.

Property ID	Property Name	Description
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 9, "Using Listening Connectors and Target Connectors," Running Integration Gateways Behind Proxy Servers, page 137.
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).
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 "+".

Copyright © 1988-2007, Oracle. All rights reserved.

- 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.

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>]]>
```

Copyright © 1988-2007, Oracle. All rights reserved.

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>
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
   </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
```

Copyright © 1988-2007, Oracle. All rights reserved.

```
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=&SubChannel=
&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⇒
&SubChannel
=SubChannel&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.

 Copyright © 1988-2007, Oracle. All rights reserved.

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/

Copyright © 1988-2007, Oracle. All rights reserved.

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 27, "Setting Up Secure Integration Environments," Implementing Nonrepudiation, page 632.

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 Chapter 12, "Sending and Receiving Messages," Handling Cookies, page 227.

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/>
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
    <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 pound sign (#). 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 pound signs. 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>
```

Copyright © 1988-2007, Oracle. All rights reserved.

```
                <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

 Copyright © 1988-2007, Oracle. All rights reserved.

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 7, “Managing Integration Gateways,” Using the integrationGateway.properties File, page 64

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.

Copyright © 1988-2007, Oracle. All rights reserved.

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 9, "Using Listening Connectors and Target Connectors," Using the HTTP Listening Connector, page 123.

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
```


Copyright © 1988-2007, Oracle. All rights reserved.

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.

See Also

Chapter 27, “Setting Up Secure Integration Environments,” Installing Web Server-Based Digital Certificates, page 603

Copyright © 1988-2007, Oracle. All rights reserved.

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:

 Copyright © 1988-2007, Oracle. All rights reserved.

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].

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
BEA WebLogic	Weblogic.jar **Note.** PeopleSoft Integration Broker currently doesn't support using the IBM WebSphere web server with a WebLogic JMS provider.
Sun iPlanet	jms.jar, ,jmq.jar, fscontext.jar, jndi.jar
IBM MQSeries	jms.jar, jndi.jar, fscontext.jar, com.ibm.mqjms.jar
Oracle Application Server (OAS)	NA

Copyright © 1988-2007, Oracle. All rights reserved.

Note. Not only can a gateway running on a BEA 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 9, "Using Listening Connectors and Target Connectors," Adding Generic JMS Providers, page 155.

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.

Property	Description
ig.jms.JMSProvider.JNDIFactory.Weblogic	Specify the JNDIFactory class name for a BEA WebLogic JMS provider. The default value is: `weblogic.jndi.WLInitialContextFactory`
ig.jms.JMSProvider.JNDIFactory. iPlanet	Specify the JNDIFactory class name for an iPlanet JMS provider. The default value is: `com.sun.jndi.fscontext.RefFSContextFactory`
ig.jms.JMSProvider.JNDIFactory. MQSeries	Specify the JNDIFactory class for an MQSeries JMS provider. The default value is: `com.sun.jndi.fscontext.RefFSContextFactory`
ig.jms.JMSProvider.JNDIFactory.OAS	Specify the JNDIFactory class name for an Oracle Application Server JMS provider. The default value is: `com.evermind.server.rmi.RMIInitialContext⇒ Factory`

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.

 Copyright © 1988-2007, Oracle. All rights reserved.

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.

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.
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.

Copyright © 1988-2007, Oracle. All rights reserved.

Property	Description
ig.jms.Queue1.Password	(Optional.) Specify the JMS queue password. If you choose to specify a password, you must encrypt it. See Chapter 7, "Managing Integration Gateways," Encrypting Passwords, page 66.
ig.jms.Queue1.SecurityPrincipal	This property is required if you are using OAS as the JMS provider. Set this property equal to the user ID for the OAS server.
ig.jms.Queue1.SecurityCredentials	This property is required if you are using OAS as the JMS provider. Set this property equal to the password to the OAS server. You must encrypt this value. See Chapter 7, "Managing Integration Gateways," Encrypting Passwords, page 66.
ig.jms.Queue1.MessageName	(Optional.) Specify the name of the service operation or message.
ig.jms.Queue1.MessageVersion	(Optional.) Specify the message version. If you specify a message name, you may also specify the message version.
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 7, "Managing Integration Gateways," Encrypting Passwords, page 66.
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.

Copyright © 1988-2007, Oracle. All rights reserved.

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 7, "Managing Integration Gateways," Encrypting Passwords, page 66.
ig.jms.Topic1.SecurityPrincipal	This property is required if you are using OAS as the JMS provider. Set this property equal to the user ID for the OAS server.
ig.jms.Topic1.SecurityCredentials	This property is required if you are using OAS as the JMS provider. Set this property equal to the password to the OAS server. You must encrypt this value. See Chapter 7, "Managing Integration Gateways," Encrypting Passwords, page 66.
ig.jms.Topic1.MessageName	(Optional.) Specify the name of the service operation or message.
ig.jms.Topic1.MessageVersion	(Optional.) Specify the message version. If you specify a message name, you may also specify the message version.
ig.jms.Topic1.RequestingNode	(Optional.) Specify the name of the requesting node.
ig.jms.Topic1.DestinationNode	(Optional.) Specify the name of the destination node.

Copyright © 1988-2007, Oracle. All rights reserved.

Property	Description
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 7, "Managing Integration Gateways," Encrypting Passwords, page 66.
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 7, "Managing Integration Gateways," Encrypting Passwords, page 66.
ig.jms.ErrorQueue.SecurityPrincipal	This property is required if you are using OAS as the JMS provider. Set this property equal to the user ID for the OAS server.
ig.jms.ErrorQueue.SecurityCredentials	This property is required if you are using OAS as the JMS provider. Set this property equal to the password to the OAS server. You must encrypt this value. See Chapter 7, "Managing Integration Gateways," Encrypting Passwords, page 66.
ig.jms.ErrorQueue-JMSFactory	Specify the queue connection factory name.
ig.jms.ErrorQueue-Url	Specify the JMS provider's URL to JNDI.

Copyright © 1988-2007, Oracle. All rights reserved.

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.
ig.jms.ErrorTopic-Password	(Optional.) Specify the JMS error topic password. If you choose to specify a password, you must encrypt it. See Chapter 7, "Managing Integration Gateways," Encrypting Passwords, page 66.
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).

Copyright © 1988-2007, Oracle. All rights reserved.

Property	Description
NodePassword	Enter the node password. This password must be encrypted. See Chapter 7, "Managing Integration Gateways," Encrypting Passwords, page 66.
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 Chapter 11, "Managing Service Operation Queues," Applying Queue Partitioning, page 199.

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.

 Copyright © 1988-2007, Oracle. All rights reserved.

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.

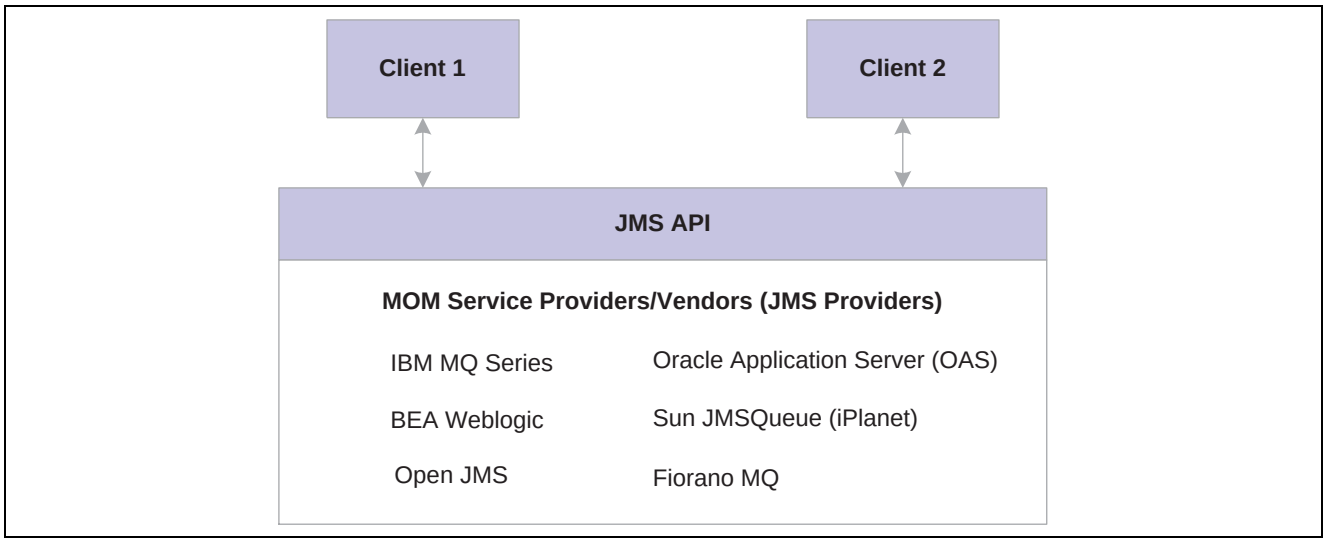
   Your web browser displays a message indicating that the JMS listening connector has stopped.

Note. You must register the JMSListeningConnectorAdministrator servlet object under the web application PSIGW in the web server. The BEA WebLogic, IBM WebSphere or OAS documentation should provide instructions about how to register the 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 Application Server, IBM MQSeries, BEA WebLogic, Sun iPlanet and others. The following diagram illustrates how messages flow through the JMS API:

Copyright © 1988-2007, Oracle. All rights reserved.



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.

Copyright © 1988-2007, Oracle. All rights reserved.

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.

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:

Copyright © 1988-2007, Oracle. All rights reserved.

Property ID	Property Name	Description
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.
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.*
JMSTARGET	JMSMessageTypeClass	(Optional.) Specify the implementation class of ProcessJMSMessage. You must set this property when the JMSMessageType is anything other than *Text*.

Copyright © 1988-2007, Oracle. All rights reserved.

Property ID	Property Name	Description
JMSTARGET	JMSPassword	(Optional.) Specify the password to access the connection. If you choose to specify a password, you must encrypt it. See Chapter 7, "Managing Integration Gateways," Encrypting Passwords, page 66.
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*. • *iPlanet*. • OracleApplicationServer
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	JMSSecurityPrincipal	Enter the user ID for the OAS server.

Copyright © 1988-2007, Oracle. All rights reserved.

Property ID	Property Name	Description
JMSTARGET	JMSSecurityCredential	Enter the password to the OAS server. This password must be encrypted. See Chapter 7, "Managing Integration Gateways," Encrypting Passwords, page 66.
JMSTARGET	JMSUrl	Specify the URL.
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).

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.

Copyright © 1988-2007, Oracle. All rights reserved.

Exception	Cause
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 BEA WebLogic, Sun iPlanet, IBM MQSeries and Oracle Application Server. 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.

Copyright © 1988-2007, Oracle. All rights reserved.

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 7, "Managing Integration Gateways," Encrypting Passwords, page 66.

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.

 Copyright © 1988-2007, Oracle. All rights reserved.

Property ID	Property Name	Description
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*.

Copyright © 1988-2007, Oracle. All rights reserved.

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.

Copyright © 1988-2007, Oracle. All rights reserved.

Property ID	Property Name	Description
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.
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 7, "Managing Integration Gateways," Encrypting Passwords, page 66.

Copyright © 1988-2007, Oracle. All rights reserved.

Property ID	Property Name	Description
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.

Copyright © 1988-2007, Oracle. All rights reserved.

Property ID	Property Name	Description
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>
```

Copyright © 1988-2007, Oracle. All rights reserved.

```
      <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 9, "Using Listening Connectors and Target Connectors," Using Directory Lists, page 161.

```
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",);
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
/* 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",);
```

Copyright © 1988-2007, Oracle. All rights reserved.

```
/* 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.

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.

 Copyright © 1988-2007, Oracle. All rights reserved.

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.

Copyright © 1988-2007, Oracle. All rights reserved.

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.

 Copyright © 1988-2007, Oracle. All rights reserved.

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 9, "Using Listening Connectors and Target Connectors," Understanding MDNs, page 164.

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:

Copyright © 1988-2007, Oracle. All rights reserved.

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.
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://pt849 //webserv//peoplesoft//keystore//pskey
ig.AS2.KeyStorePassword	Specify the encrypted password to the Java keystore. For example: GD9klUFw8760HVaqeT4pkg==

Copyright © 1988-2007, Oracle. All rights reserved.

Property	Description
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
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.

Copyright © 1988-2007, Oracle. All rights reserved.

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.
AS2PROPERTY	AsynchronousMDN RecipientURL	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.

Copyright © 1988-2007, Oracle. All rights reserved.

Property ID	Property	Description
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.
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*.

Copyright © 1988-2007, Oracle. All rights reserved.

Property ID	Property	Description
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.
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.

Copyright © 1988-2007, Oracle. All rights reserved.

Property ID	Property	Description
PRIMARYURL	URL	Specify the URL to which messages are sent using this connector.
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: <PS_HOME>\webserver\peoplesoft\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://pt849//webserv//peoplesoft//keystore//pskey
ig.AS2.KeyStorePassword	Specify the encrypted password to the Java keystore. For example: GD9klUFw8760HVaqeT4pkg==

Copyright © 1988-2007, Oracle. All rights reserved.

Property	Description
ig.AS2.AS2Directory	Specify the directory to log MDN responses. This property is required for asynchronous MDNs. For example: c://temp//as2
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

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*.

Gateway-Level 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
```

Node-Level Connector Properties

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

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.

Copyright © 1988-2007, Oracle. All rights reserved.

Property ID	Property Name	Description
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.

Copyright © 1988-2007, Oracle. All rights reserved.

 Copyright © 1988-2007, Oracle. All rights reserved.

CHAPTER 10

Managing Messages

This chapter provides an overview of managing messages and discusses how to:

- Create message definitions.
- Manage rowset-based messages.
- Manage nonrowset-based messages.
- Manage message parts.
- Manage container messages.
- Rename and delete message definitions.
- Delete messages during upgrade.

Understanding Managing Messages

This section provides an overview of messages.

Message Definitions

Message definitions provide the physical description of the data that is being sent, including fields, field types, and field lengths. You create message definitions in the PeopleSoft Internet Architecture.

Note. Messages are shapes that describe the contents of a service operation transaction. Unlike prior PeopleTools releases, messages do not contain any processing logic. All processing logic is defined in service operations, using service operation handlers.

Message Types

Four types of messages are available:

Rowset-based messages For hierarchical data that is based on PeopleSoft records, you create a message definition by assembling records, organizing them into a hierarchy, and selecting fields from those records to include in the message. The result is a rowset that doesn't need to match an existing rowset structure in the application. Use the PeopleCode Rowset and operation classes to generate, send, receive, and process these messages.

Nonrowset-based messages These messages can have virtually any structure and content. You create a message definition, but you do not insert any records. The message definition serves as a placeholder for the actual message. Use the PeopleCode XmlDoc and operation classes to generate, send, receive, and process these messages. If

Copyright © 1988-2007, Oracle. All rights reserved.

you're handling Simple Object Access Protocol (SOAP) compliant data, you can also use the SoapDoc class to generate and process these messages.

Container messages A container message is a nonrowset-based message that holds one or more part messages.

A container message must contain all rowset-based messages *or* all nonrowset-based message parts.

Message parts Message parts are rowset-based messages or nonrowset-based messages that you designate as a part message, to be used in a container message.

The following table describes when to use a given message type:

Message Type	When to Use
Rowset-based message.	All PeopleSoft-to-PeopleSoft integrations.
Nonrowset-based message.	Integrations with third-party systems.
Container message with rowset-based message parts.	Exposing PeopleSoft services to third-party systems. (Nested message parts not supported.)
Container message with nonrowset-based message parts.	Exposing PeopleSoft services to third-party systems and need to provide nested parts.

Message Record Structure

If a message handles PeopleSoft record data, that is, a rowset-based message, you must insert records in the message definition in an appropriate hierarchical structure.

However, if the message data doesn't map to a record hierarchy, do not insert any records. You supply or receive the data and its structure from an external source, using the PeopleCode XmlDoc or SoapDoc classes.

See Chapter 12, "Sending and Receiving Messages," page 205.

Underlying Record Definitions

Records that you insert in a message definition are proxies for the original record definitions. Any change that you make to an underlying record definition is automatically reflected by a change in the corresponding record in the message definition.

Fields Defined as Uppercase

If a message definition includes character fields that are defined as uppercase, then character data in those fields is automatically converted to uppercase when the message is received. This happens when the receiving PeopleCode inserts the message data in a rowset, and it overrides any previous changes in the data, including transformation and data translation.

Restrictions for Modifying Messages

Messages may become restricted and become read-only under the following conditions:

- The service to which a message is tied is a restricted service.

 Copyright © 1988-2007, Oracle. All rights reserved.

- If a message is used internally by the system. For example, the delivered IB_GENERIC message is read-only and is used internally by the system.
- A message is referenced in the runtime tables.

  To work around this, you must remove any entries in the runtime tables that reference the message.
- If a message is a message part.
- If a message is used in a service operation where WSDL documents have been generated.
- If a message is used in a service operation that has validation enabled.

Adding Message Definitions

This section discusses how to add a message definition to the system.

Understanding Adding Message Definitions

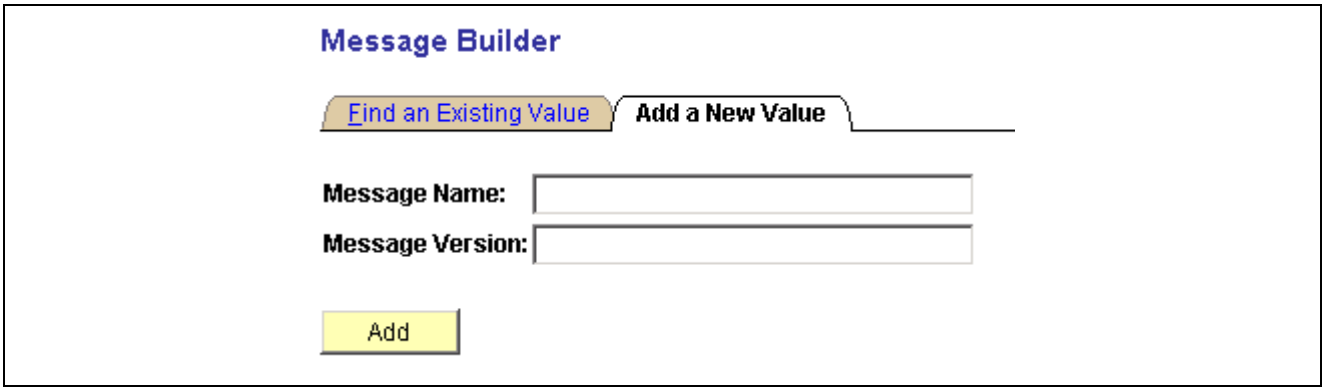
When you create a message definition, you first create the message definition, in which you name the message and specify the message version. After doing so, you can then define additional aspects of the message definition.

Page Used to Add Message Definitions

Page Name	Object Name	Navigation	Usage
Message Builder page	IB_MESSAGE_BUILDER	Select PeopleTools, Integration Broker, Integration Setup, Messages.	Add message definitions and configure message definition.

Adding a Message Definition

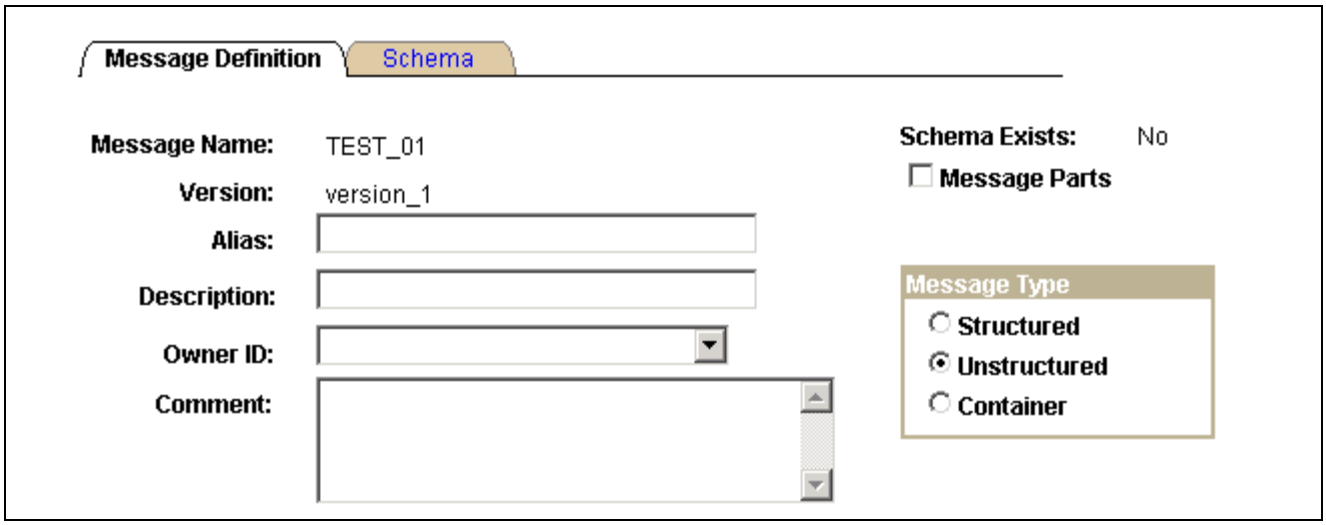
The following example shows the Message Builder page that you use to name a new message definition and assign a version to it:



Message Builder - Add a New Value page

The following example shows the Messages - Message Definition page that you use to configure a message after you create the message definition:

Copyright © 1988-2007, Oracle. All rights reserved.



Messages - Message Definition page

Note. For asynchronous integrations, define a single message. For synchronous integrations, define two messages: one request message and one response message, unless the request and response have the same shape.

To add a message definition:

1. Select PeopleTools, Integration Broker, Integration Setup, Messages.
2. Select the Add New Value tab.
3. In the Message Name field, enter a name for the message.

   The message name cannot exceed 30 characters. Do not include any spaces in the message name.
4. In the Message. Version field, enter a version for the message.

   The message version cannot exceed 30 characters. Do not include any spaces in the message version.

   Accepted formats for the message version include:

   - *Version_1*.
   - *V1*.
5. Click the Add button.

   The Messages - Message Definition page appears.
6. In the Message Type group box, select the message type. Values are:

   - *Rowset-based*
   - *Nonrowset-based* (default)
   - *Container*
7. (Optional) In the Alias field, enter the name that the external system is expecting, if different from the value in the Message Name field.

   This field appears only when you are defining nonrowset-based or container messages.
8. (Optional) Select the Message Parts check box if the message will be used as a message part in a container message definition.
9. (Optional) In the Description field, enter a description for the definition.

Copyright © 1988-2007, Oracle. All rights reserved.

10. (Optional) From the Owner ID dropdown list box, select an owner for the definition.

    The owner ID helps to determine the application team that last made a change to the definition. The values in the dropdown list box are translate table values that you can define in the OBJECTOWNERID field record.

11. (Optional) In the Comment field, enter any pertinent comments about the definition.

12. The next step depends on the type of message definition that you are creating:

    - *Rowset-Based Message.* You must add a root record to the definition before you can save it.

      See Chapter 10, "Managing Messages," Managing Rowset-Based Messages, page 181.

    - *Nonrowset-Based Message.* The message definition is complete and you can click the Save button to save the changes. You can now add an XML message schema to the definition.

      See Chapter 10, "Managing Messages," Managing Nonrowset-Based Messages, page 186.

    - *Container Message.* You must add at lease one message part to the definition before you can save the changes.

      See Chapter 10, "Managing Messages," Managing Container Messages, page 189.

Managing Rowset-Based Messages

This section provides an overview of managing rowset-based message definitions and discusses how to:

- Insert root records.
- Insert child and peer records.
- Specify record aliases.
- Delete records.
- Exclude fields from messages.
- Specify field name aliases.
- Generate XML message schemas for rowset-based messages.

Understanding Managing Rowset-Based Messages

This section provides overview information about managing rowset-based message definitions.

Root Records

When you create a rowset-based message, you must at a minimum insert a root record (level 0) into the definition.

Records and Record Fields

You create and modify records and record fields in PeopleSoft Application Designer.

Note. Avoid using work records in messages. Work records don't behave like regular records when used with PeopleCode rowset methods. A good alternative is to use dynamic views.

Copyright © 1988-2007, Oracle. All rights reserved.

Record and Record Field Aliases

Record and field aliases are optional parameters that are used for schema and XML generation.

When record and field aliases are used, the receiver of a message sees the alias names instead of the actual record and field names. The alias names are seen in the message definition, in message schemas, and on generated runtime XML that is sent to the receiver.

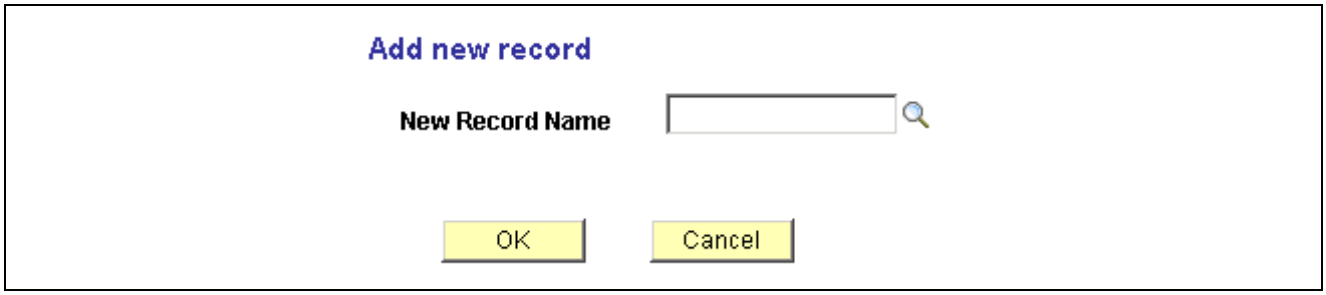
Note that the sender still codes to the actual record and field name.

Pages Used to Manage Rowset-Based Messages

Page Name	Object Name	Navigation	Usage
Add New Record page	IB_MESSAGE_TOP_SEC	Select PeopleTools, Integration Broker, Integration Setup, Messages. The Message Definitions page appears. Click the Add Record to Root link.	Insert a root record into a rowset-based message definition.
Message Record Properties page	IB_MESSAGE_REC_SEC	From the Message Definitions page, click the hyperlinked name of a record.	Insert child and peer records to a record, specify record aliases, delete records, and exclude fields from records.
Message Field Properties page	IB_MESSAGE_FLD_SEC	From the Message Definitions page, click the hyperlinked name of a field.	Specify a field name aliases. Include or exclude the field from the message definition.
Schema page	IB_MESSAGE_BUILD2	From the Message Definitions page, click the Schema tab.	Generate or delete XML schema for a rowset-based message.

Inserting Root Records

You insert a root record into a rowset-based message definition using the Add a New Record page shown in the following example:



Add New Record page

Note. There can only be one root record defined for each rowset-based message.

To insert a root record into a definition:

1. On the Messages–Message Definitions page, click the Add Record to Root link.

   The Add New Record page appears.

 Copyright © 1988-2007, Oracle. All rights reserved.

2. In the New Record Name field, enter the name of the record to add, or click the Lookup button to search for and select one.
3. Click the OK button.

The root record appears in the tree structure. Click the plus button to expand the tree and view fields that are associated with the record.

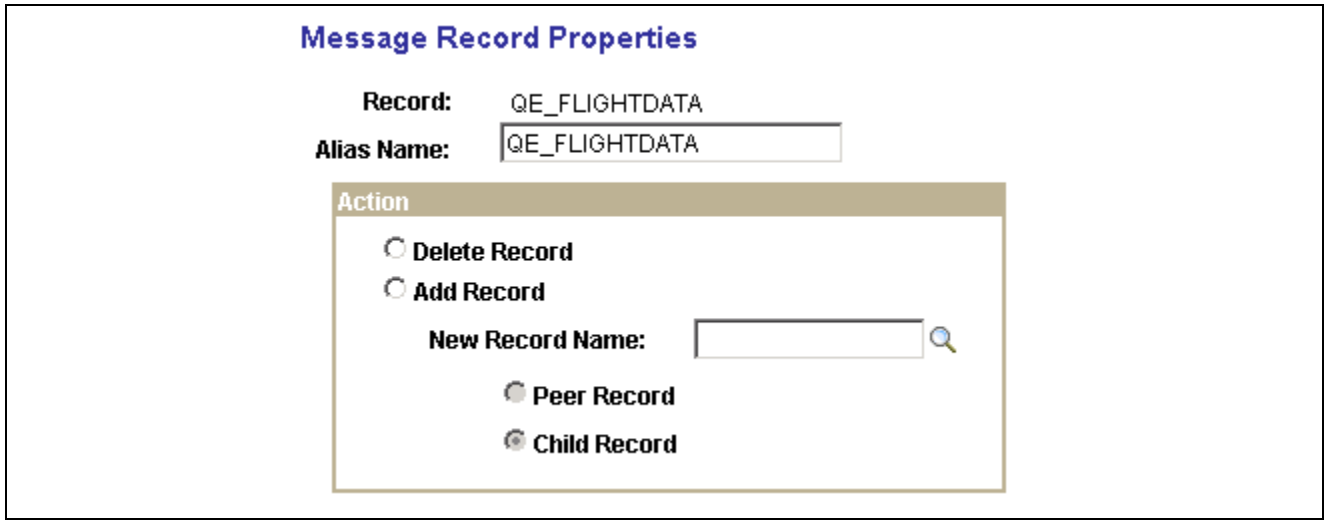
You can exclude fields from the record and specify field name aliases. Steps for performing these actions are described elsewhere in this chapter.

See Chapter 10, "Managing Messages," Excluding Fields from Messages, page 184.

See Chapter 10, "Managing Messages," Specifying Field Name Aliases, page 185.

Inserting Child and Peer Records

You insert child and peer records into a rowset-based message definition using the Message Record Properties page shown in the following example:



Message Record Properties page

To insert a child or peer record into a rowset-based message definition:

1. On the Messages–Message Definition page, click the linked record name to which to add a peer or child record.

   The Message Record Properties page appears.
2. In the Action group box, select Add Record.
3. In the New Record Name field, enter the name of the record to add, or click the Lookup button to search for and select a name.
4. Select whether to add the record as a peer record or a child record.
   - Select Peer Record to add the record as a peer.
   - Select Child Record to add the record as a child.
5. Click the OK button.

   The Messages–Message Definitions page appears.
6. Click the Save button.

Copyright © 1988-2007, Oracle. All rights reserved.

Specifying Record Aliases

You can specify aliases of the root, peer, and child records that you insert into rowset-based messages.

To specify a record alias:

1. In the Messages–Message Definitions page, click the name of the record for which you want to specify an alias.

   The Message Record Properties page appears.
2. In the Alias Name field, enter an alias name.
3. Click the OK button.

   The Messages–Message Definitions page appears.
4. Click the Save button.

Deleting Records

This section describes how to delete records from a rowset-based message.

Note. Deleting the root record deletes all records and their associated fields that are inserted into the definition.

To delete a record:

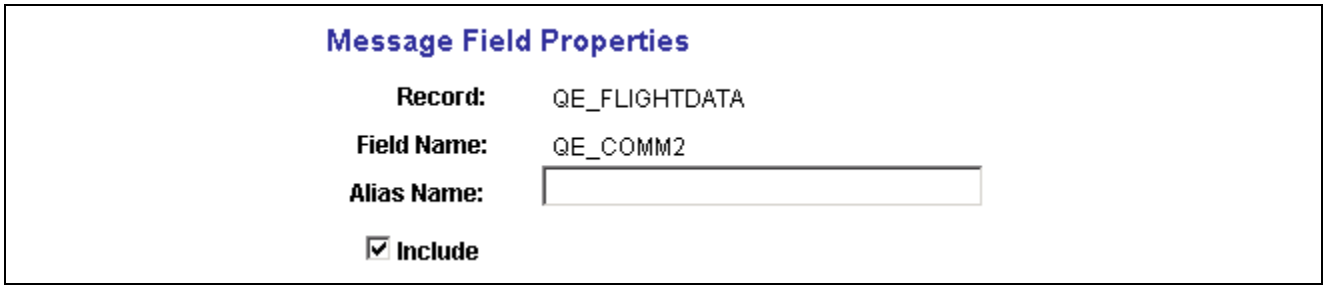
1. In the Messages–Message Definitions page, click the name of the record to delete.

   The Message Record Properties page appears.
2. In the Action group box, select Delete Record.
3. Click the OK button.

   The Messages–Message Definitions page appears.
4. Click the Save button.

Excluding Fields from Messages

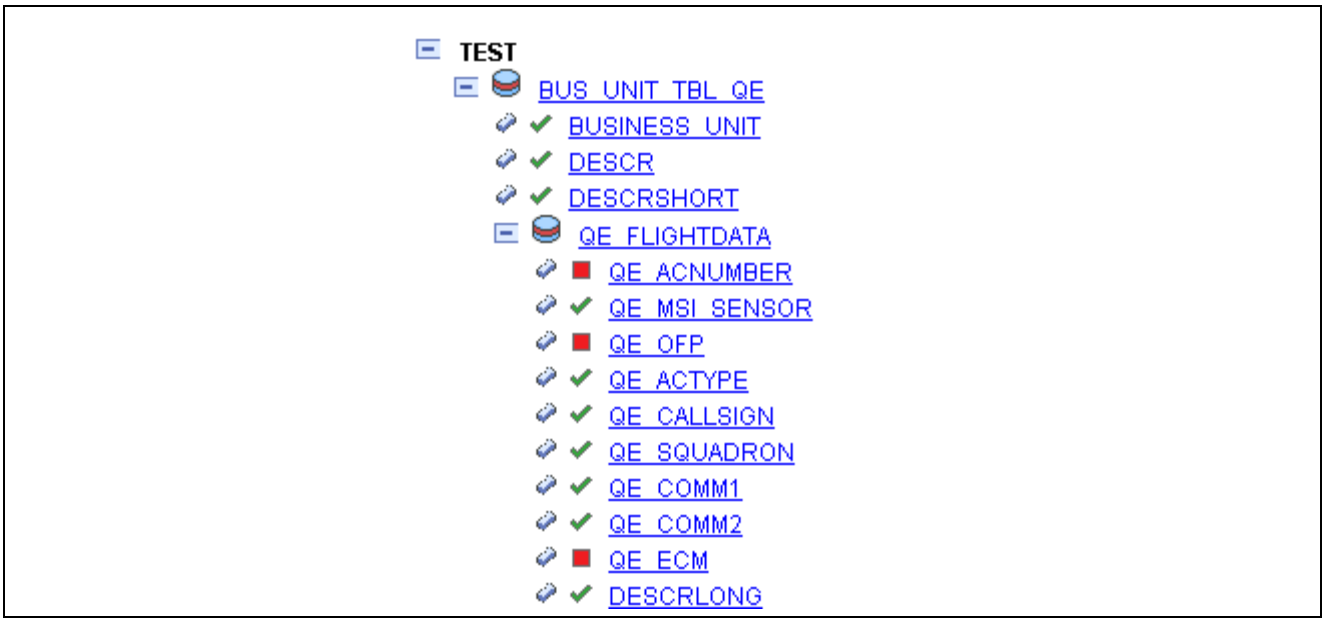
You can exclude record fields from inclusion in a rowset-based message definition using the Message Field Properties page shown in the following example:



Message Field Properties page for the field QE_COMM2

After you exclude fields from records, the tree view of the message definition on the Message Definitions page displays a red icon next to the excluded fields. The following example show two fields, *QE_ACNUMBER* and *QE_OFP*, have been excluded from the *QE_FLIGHTDATA* record.

 Copyright © 1988-2007, Oracle. All rights reserved.



Fields excluded from the QE_FLIGHTDATA record

To exclude fields:

1. From the Messages–Message Definitions page, click the plus button to expand the record tree structure, and locate the field to exclude from the definition.
2. Click the name of the field to exclude.

   The Message Field Properties page appears.
3. Clear the Include check box.
4. Click the OK button.

   The Messages–Message Definitions page appears.
5. Click the Save button.

Specifying Field Name Aliases

To specify a field name alias:

1. From the Messages–Message Definitions page, click the plus button to expand the record tree structure, and locate the field to exclude from the definition.
2. Click the name of the field to exclude.

   The Message Field Properties page appears.
3. In the Alias Name field, enter an alias name.
4. Click the OK button.

   The Messages–Message Definitions page appears.
5. Click the Save button.

Generating XML Message Schemas for Rowset-Based Messages

This section discusses how to:

Copyright © 1988-2007, Oracle. All rights reserved.

- Build XML message schemas for rowset-based messages.
- Delete XML message schemas for rowset-based messages.

Building XML Message Schemas for Rowset-Based Messages

After you create a rowset-based message definition and insert a root record into the definition, and optionally, additional peer and child records, you can generate an XML message schema for the definition.

Note. You cannot regenerate message schemas for messages that are defined as part of a restricted service.

To generate an XML message schema for a rowset-based message:

1. Select PeopleTools, Integration Broker, Integration Setup, Messages.
2. Select the rowset-based definition for which to generate an XML schema.

   The Messages–Message Definitions page appears.
3. Click the Schema tab.
4. Click the Build Schema button.

The results appear at the bottom of the page.

Clicking the Build Schema button saves the schema and the message.

Note that when you click the Message Definitions tab to view the definition, the Schema Exists field displays a value of *Yes,* indicating that a schema has been generated for the definition.

Deleting XML Message Schemas for Rowset-Based Messages

To delete an XML message schema for a rowset-based message:

1. Select PeopleTools, Integration Broker, Integration Setup, Messages.
2. Select the rowset-based definition for which to generate an XML schema.

   The Messages - Message Definitions page appears.
3. Click the Schema tab.
4. Click the Delete Schema button.

The results appear at the bottom of the page.

Note that when you click the Message Definitions tab to view the definition, the Schema Exists field displays a value of *No,* indicating that a schema has been deleted for the definition.

Managing Nonrowset-Based Messages

This section provides an overview of managing nonrowset-based messages and discusses how to:

- Add XML message schemas to nonrowset-based messages.
- Edit nonrowset-based XML message schemas.

 Copyright © 1988-2007, Oracle. All rights reserved.

Understanding Managing Nonrowset-Based Messages

After you create a nonrowset-based message definition, you do not need to complete any additional configuration steps for the definition, except to add an XML schema. The XML schema describes the data to be sent, and includes the field names, data types, field lengths and so on.

See Also

Chapter 10, "Managing Messages," Adding Message Definitions, page 179

Page Used to Manage Nonrowset-Based Messages

Page Name	Object Name	Navigation	Usage
Schema page	IB_MESSAGE_BUILDER	Select PeopleTools, Integration Broker, Integration Setup, Messages. The Message Definitions page appears. Click the Schema tab.	Add and edit XML schemas for nonrowset-based messages

Adding XML Message Schemas to Nonrowset-Based Messages

To add an XML message schema to nonrowset-based messages:

Note. You cannot regenerate message schemas for messages that are defined as part of a restricted service.

1. Select PeopleTools, Integration Broker, Integration Setup, Messages.
2. Select the nonrowset-based definition to which you want to add an XML schema.

   The Messages - Message Definitions page appears.
3. Click the Schema tab.
4. Click the Add Schema button.

   The Schema page appears.
5. Add the XML schema to the message.

   You can add the schema to the message in two ways:

   - Click the Upload Schema From File button to browse for and upload a schema that you have already saved to a file.
   - Enter the XML schema in the Schema text box that is provided.
6. Click the Save button.

If you define the message as a message part, you must supply a schema to save the message. All message parts require a schema at save time.

Editing Nonrowset-Based XML Schemas

After an XML message schema is added to a nonrowset-based message, you can edit the schema using the Schema page.

Note. You cannot regenerate message schemas for messages that are defined as part of a restricted service.

Copyright © 1988-2007, Oracle. All rights reserved.

To edit nonrowset-based XML message schemas:

1. Select PeopleTools, Integration Broker, Integration Setup, Messages.
2. Select the nonrowset-based definition that contains the schema that you want to edit.

   The Messages - Message Definitions page appears.
3. Click the Schema tab.

   The Schema page appears and displays the existing XML message schema for the definition.
4. Click the Edit Schema button.
5. In the Schema text box, make your changes and additions to the XML schema.
6. Click the Save button.

Managing Message Parts

This section discusses how to create message parts.

Understanding Message Parts

Message parts are individual message definitions that get used in container messages.

While they can be rowset-based or nonrowset-based, the advantage of using message parts comes when working with rowset-based messages.

Container messages are always nonrowset-based. So, if you use a container message that contains rowset-based part messages, the container messages sends XML that contains none of the standard PeopleSoft message XML structures, such as PSCAMA, FieldTypes, and so on. However, you can use the rowset-based classes and methods to populate and read the structure of each part message.

Creating Part Messages

To create a part message, create a standard rowset-based or nonrowset-based message and click the Part Message box on the Message Definition page.

When the service system status is set to *Production*, once a message is used in a container message, you cannot alter the message while it is associated with a container message.

You must generate schemas for all part messages before you can save them.

Schemas for rowset-based messages are automatically built when the message is saved. Schemas for nonrowset-based parts must be added in order to save the message.

See Also

Chapter 10, “Managing Messages,” Adding Message Definitions, page 179

Chapter 10, “Managing Messages,” Managing Container Messages, page 189

 Copyright © 1988-2007, Oracle. All rights reserved.

Managing Container Messages

This section provides an overview of managing container messages and discusses how to:

- Add message parts to container messages.
- Generate XML message schemas for container messages.

Understanding Managing Container Messages

Container messages are used for those situations where you want to produce XML that contains none of the standard PeopleSoft messaging XML structures , such as PSCAMA, FieldType, and so on, yet you want to use PeopleSoft rowset-based classes and methods to populate and read the message structure.

Container messages hold one or more message parts.

Container messages are always nonrowset-based messages.

The message parts you add to a container message must all be rowset-based message parts, or all nonrowset-based message parts.

Pages Used to Manage Container Messages

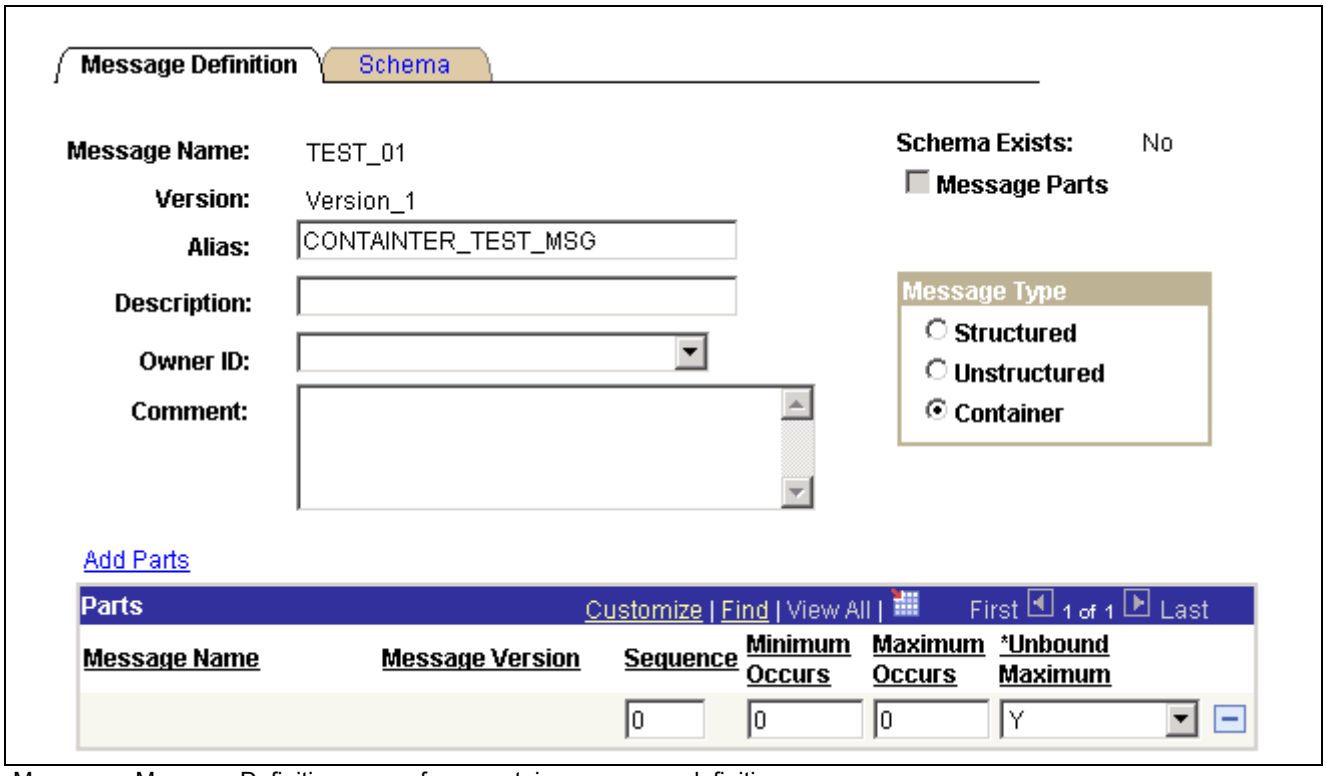
Page Name	Object Name	Navigation	Usage
Message Definition page	IB_MESSAGE_BUILDER	Select PeopleTools, Integration Broker, Integration Setup, Messages.	Add message parts to container messages.
Schema page	IB_MESSAGE_BUILD2	Select PeopleTools, Integration Broker, Integration Setup, Messages. The Message Definitions page appears. Click the Schema tab.	Generate XML schemas for container messages

Adding Message Parts to Container Messages

This section discusses how to add message parts to container messages.

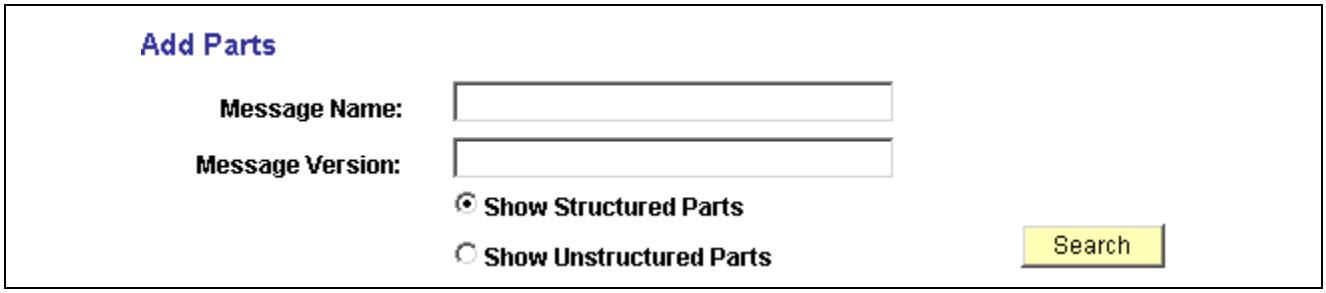
Use the Messages–Message Definitions page to perform this task. To access the page, select PeopleTools, Integration Broker, Integration Setup, Messages. The following example shows this page:

Copyright © 1988-2007, Oracle. All rights reserved.



Messages–Message Definitions page for a container message definition

When you click the Add Parts link to specify a message, version, and message type to add, the Add Parts page appears as shown in the following example:

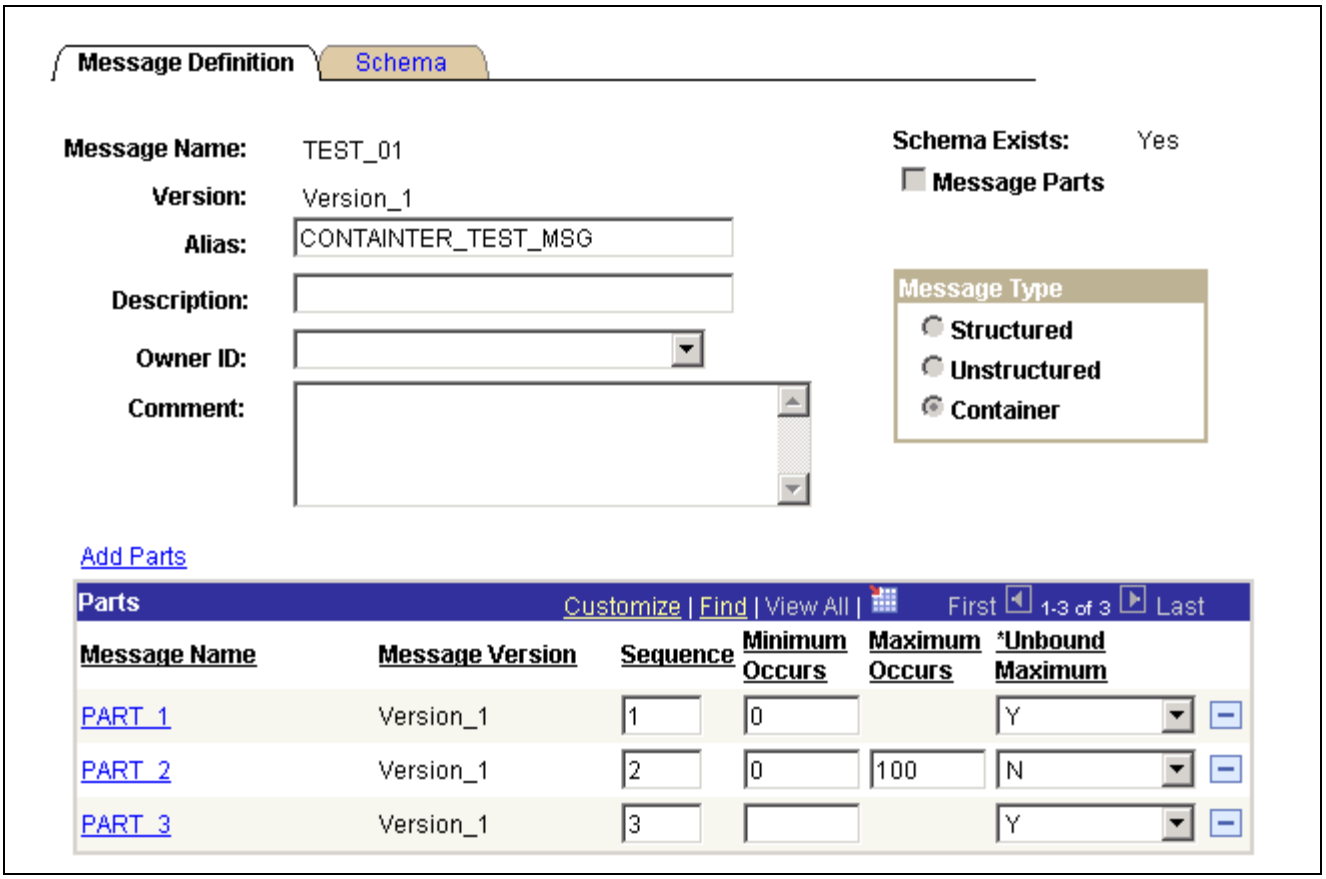


Add Parts page

For a message definition to be available for you to add to a container message, you must have selected the Message Parts check box when you created the message definition. In addition, container messages can contain only all rowset-based messages or all nonrowset-based messages.

After you add message parts to a container message, the Messages–Message Definitions page displays and the message parts that you have added to the message are listed in the Parts grid. The following examples show of three message parts that are added to a container message:

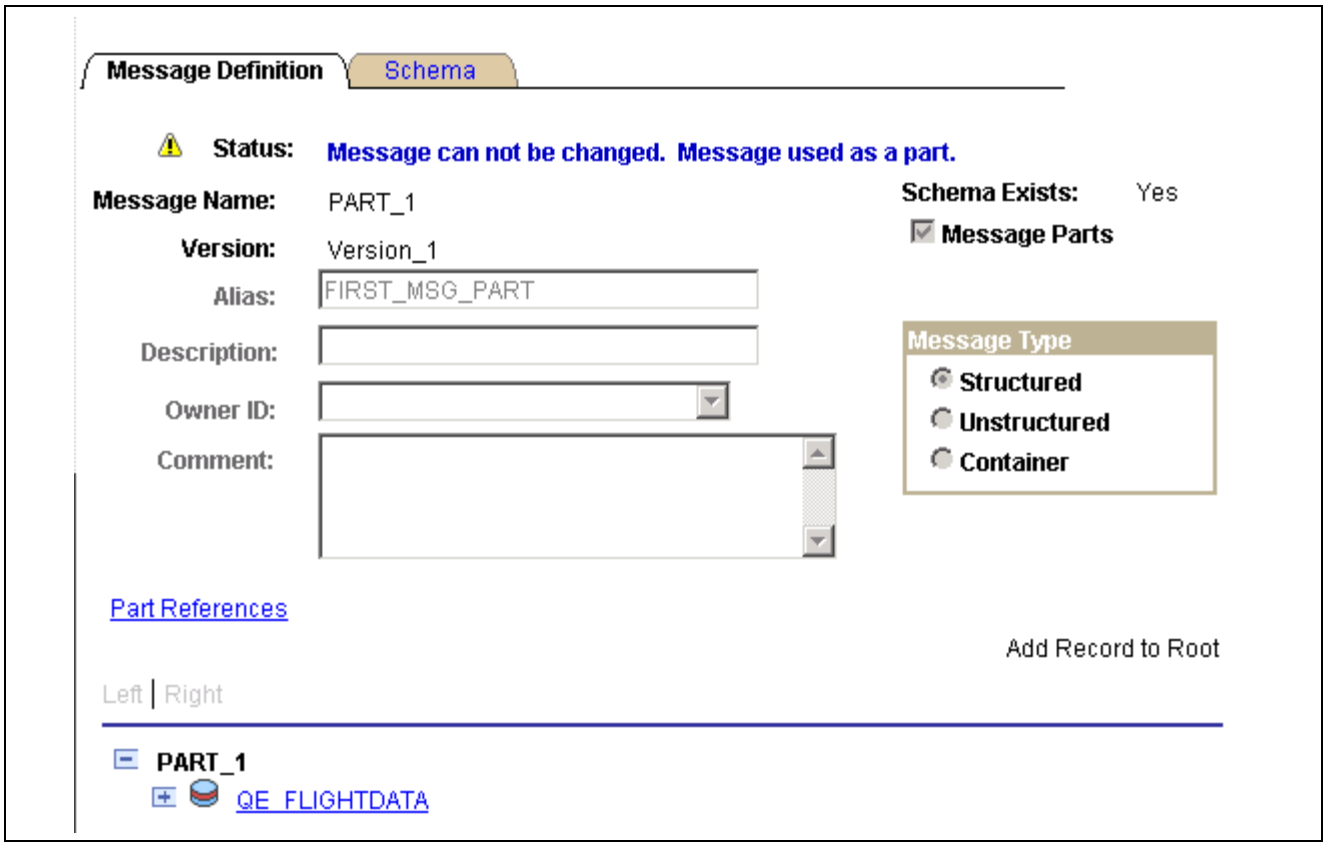
Copyright © 1988-2007, Oracle. All rights reserved.



A container message that contains three message parts

Click the name of any of the message parts that appear in the grid to open the individual message definition. If the service system status is set to *Production*, when assigned to a container message, you cannot modify a message definition. To modify the definition, you must delete it from the container message first. The following example shows how the PART_1 message part displays if you click the message name in the Parts grid:

Copyright © 1988-2007, Oracle. All rights reserved.



PART_1 message definition

Clicking the Part References link displays all messages to which the message part is assigned.

Note. Before you add nonrowset-based message parts to a container message, you must upload XML message schemas to each message part that you intend to include in the container message. Nonrowset-based part messages cannot be saved without a schema.

To add a message part to a container message:

1. From the Messages–Message Definitions page, click the Add Parts link.

   The Add Parts page appears.

2. Select a message to add.

   You can use one of two methods to select a message to add:

   a. In the Message Name and Message Version fields, enter the message name and version to add.

   b. Select the Show Rowset-Based Parts option or the Show Nonrowset-Based Part option and click the Search button to display all rowset-based or nonrowset-based messages that are designated as part messages in the system.

      Select one or more messages to include in the container message.

3. Click the OK button.

   The Messages–Message Definitions page appears, with the Parts grid populated with the message or messages that you selected.

4. (Optional.) In the Parts grid, enter numeric values in the Sequence column to order message part placement in the container message.

Copyright © 1988-2007, Oracle. All rights reserved.

If you do not enter any values, the system sequences the messages in the order in which you add them to the container message.

5. (Optional.) Specify the minimum or maximum rows that are required to have a valid message.
    a. (Optional.) In the Minimum Occurs field, enter the number of minimum rows in the message part to include in the container message.
    b. (Optional.) To specify a maximum number of rows from the message part to include in the container message, from the Unbound Maximum dropdown list box, select *N*. The Maximum Occurs field appears in the grid.

       In the Maximum Occurs field, enter the maximum number of rows from the part message to include in the container message.

The default value for the Unbound Maximum field is *Y*. This default value means that the number of rows from the part message that the system includes in the container messages is unlimited, or unbound. When you select the default, all rows from a part message are included in the container message. If you change the field value to *N,* you must enter the maximum number of rows from the part message to include in the container message in the Maximum Occurs field.

Generating XML Message Schemas for Container Messages

XML message schemas for container messages re automatically generated when you save the definition. You can view the generated XML message schema on the Messages–Schema page. The following example shows this page:

Copyright © 1988-2007, Oracle. All rights reserved.

System-generated XML message schema for container message with rowset-based message parts

The namespace that is used in the XML message schema becomes by default the value that is defined on the Service Configuration page. To change the namespace, enter a the new namespace on the Schema page in the Namespace field, the Message Definition tab, and save the change. The XML message schema is generated again by means of the modified namespace value.

Renaming and Deleting Message Definitions

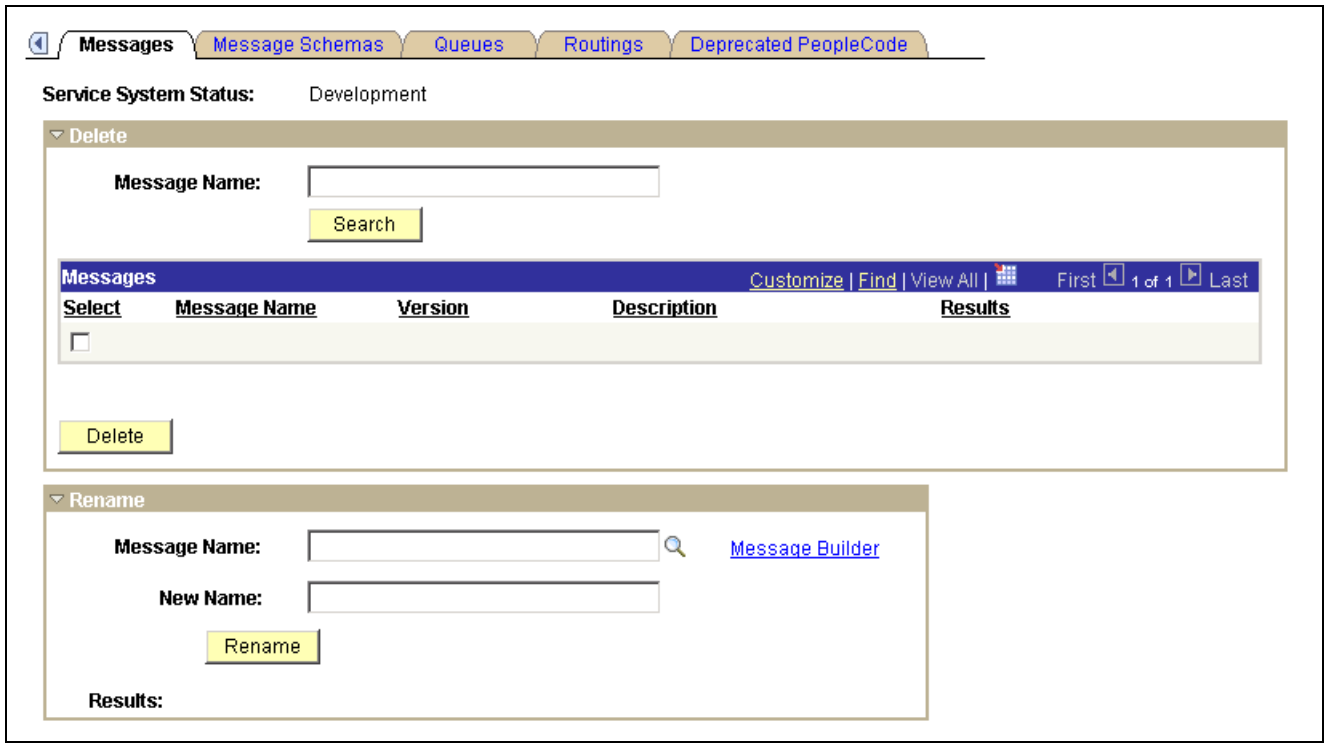
You can rename and delete messages using the Messages page in the Services Administration component (IB_HOME_PAGE). The Message page contains two sections: a Delete section that enables you to delete message definitions and a Rename section that enables you to rename message definitions.

To access the page, select PeopleTools, Integration Broker, Service Utilities, Service Administration, and click the Messages tab.

When you first access the Messages page, all sections are collapsed. Click the section header arrow buttons to expand and collapse each section.

The following example shows the Messages page with the Delete and Rename sections expanded:

Copyright © 1988-2007, Oracle. All rights reserved.



Services Administration – Messages page with the Delete and Rename sections expanded

At the top of the page, the Service System Status field displays the current setting. The service system status, set on the Service Configuration page, affects the ability to rename and delete messages.

See Chapter 14, "Managing Services," Understanding Configuring PeopleSoft Integration Broker for Handling Services, page 277 and Chapter 14, "Managing Services," Setting Service Configuration Properties, page 279.

Pages Used to Rename and Delete Message Definitions

Page Name	Object Name	Navigation	Usage
Service Administration-Messages page	IB_HOME_PAGE5	Select PeopleTools, Integration Broker, Service Utilities, Service Administration. Click the Messages tab.	Rename and delete message definitions.

Renaming Message Definitions

To rename a message definition:

Note. Renaming a message definition renames all versions.

1. Access the Services Administration – Messages page.
2. Click the arrow next to the Rename section header to expand the section.
3. In the Message Name field, enter the message definition to rename, or click the Lookup button to search for and select the message to rename.
4. (Optional.) Click the Message Builder link to view details about the selected message in the Messages–Message Definitions page.

Copyright © 1988-2007, Oracle. All rights reserved.

When you are done viewing the message details, click the Return button to return to the Services Administration –Messages page.

5. In the New Name field, enter the new name for the message definition.
6. Click the Rename button.

Deleting Messages Definitions

To delete a message definition:

1. Access the Services Administration –Messages page.
2. Click the arrow next to the Delete section header to expand the section.
3. In the Message Name field, enter the name of the message to delete, and click the Search button.

   Search results appear in the results grid.
4. In the results grid, select the check box next to the message or messages to delete.
5. Click the Delete button.

Deleting Messages During Upgrade

To delete a message definition in an application upgrade project, you must first make sure that no live instances of the message exist. Archive or delete any such messages in both the source and the target database. Otherwise, you receive an error message during the copy process indicating that the object is in use.

 Copyright © 1988-2007, Oracle. All rights reserved.

CHAPTER 11

Managing Service Operation Queues

This chapter discusses how to:

- Add queue definitions.
- Apply queue partitioning.
- Set permissions to queues.
- Rename and delete queue definitions.
- Delete queues during upgrade.

Understanding Service Operation Queues

Service operations queues are used to queue service operations for processing.

Note. Beginning with PeopleTools 8.48 queues replace message channels from previous PeopleTools 8.4x releases.

Adding Queue Definitions

This section discusses how to create queue definitions.

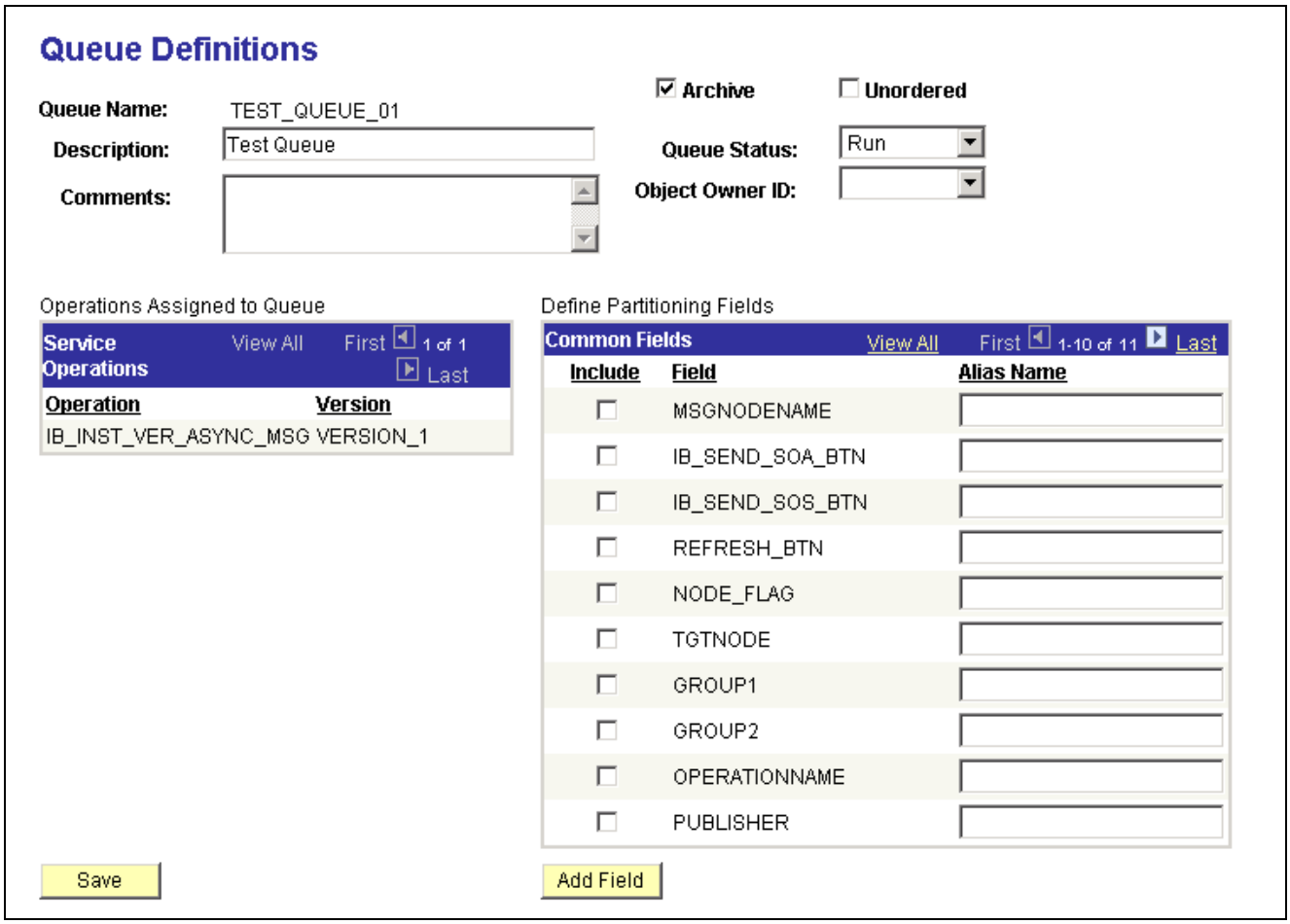
Page Used to Create Queue Definitions

Page Name	Object Name	Navigation	Usage
Queue Definitions page	IB_QUEUEDEFN	Select PeopleTools, Integration Broker, Integration Setup, Queues.	Create queue definitions.

Adding a Queue Definition

The following example shows the Queue Definitions page.

Copyright © 1988-2007, Oracle. All rights reserved.



Queue Definition page

You work with the following page elements when you add a queue.

Queue Name Enter the name of the queue.

Archive Select to archive service operation instances that are assigned to the queue. By default, archiving is enabled.

If you clear this check box, the messaging archive process purges the queue entries that have been processed.

This check box also controls whether the Archive or Delete action is available in the Asynchronous Details component of the Services Operations Monitor.

Unordered Select to enable field partioning and to process service operations unordered.

By default, the check box is cleared and inbound service operations that are assigned to a queue are processed one at a time sequentially in the order that they are sent.

Select to force the channel to handle all of its service operations in parallel (unordered), which does not guarantee a particular processing sequence. This disables the channel's partitioning fields.

Clear this check box if you want all of the queues's service operations processed sequentially or if you want to use the partitioning fields.

Copyright © 1988-2007, Oracle. All rights reserved.

See Chapter 11, "Managing Service Operation Queues," Applying Queue Partitioning, page 199.

Description Enter a description for the queue.

Queue Status Values are:

Run: (default) Service operations that are assigned to this queue are received and processed normally.

Pause: Service operations are received but not processed until the status is reset to *Run.*

Note. You can also pause and restart queues in the Service Operations Monitor.

See Chapter 21, "Using the Service Operations Monitor," Pausing and Starting Queues, page 488.

Object Owner ID From the dropdown list box, select the object owner.

The owner ID helps to determine the application team that last made a change to the definition. The values in the dropdown list box are translate table values that you can define in the OBJECTOWNERID field record.

Comments Use this area to enter comments about the definition.

Operations Assigned to Queue This read-only section lists all service operations that are assigned to the queue.

Include Select the Include check box next to a field name to include the field in queue partitioning.

See Chapter 11, "Managing Service Operation Queues," Applying Queue Partitioning, page 199.

Add Field Click to view and select partitioning fields.

See Chapter 11, "Managing Service Operation Queues," Applying Queue Partitioning, page 199.

Applying Queue Partitioning

This section provides an overview of queue partitioning and discusses how to select partitioning fields.

Understanding Queue Partitioning

By default, all inbound service operations that are assigned to a given service operation queue are processed one at a time, in the order they are sent. Consequently, the sending node can engage in a series of transactions that modify a specific record, and the changes are applied by the receiving node in the correct order. This can be inefficient if the sequence does not matter or if the sequence is relevant only within groups of service operations with the same key values. One solution to this inefficiency is partitioning.

Note. Partitioning applies only to asynchronous messaging.

Copyright © 1988-2007, Oracle. All rights reserved.

To maximize messaging efficiency and throughput, you want the system to simultaneously process as many service operations as possible. Because queues enforce service operation sequence, ideally you have a separate service operation queue for each group of service operations that must be processed in order. You can achieve this goal by designating specific fields that are common to the service operations that are assigned to a queue. These fields partition, or divide, the queues into subqueues. PeopleSoft Integration Broker creates these subqueues at runtime.

Each subqueue processes only the service operations for which the designated common fields have an identical combination of values. The service operations within each subqueue are processed in the order that they are sent, so they remain in sequence. Each subqueue works in parallel with the other subqueues—all subqueues simultaneously process their own associated service operations.

You implement partitioning by designating the partitioning fields in the queue definition; no other steps are required.

Note. The more partitioning fields that you designate, the more subqueues are generated. If you designate a combination of fields that are unique for each service operation, all service operations are processed simultaneously, in their own subqueues, without regard to sending order. This is the equivalent of selecting the Unordered check box in the queues definition.

Selecting Partitioning Fields

You can partition queues using any combination of:

- Database record fields.
- Message header fields.
- Message XML fields.

Note. Fields of types image and long character aren't available for partitioning. In addition, when you are partitioning nonrowset-based or inbound messages, tags cannot be mixed case.

Database Record Fields

Database record fields are the data fields in a PeopleSoft rowset-compatible message. Typically, the more service operations that you assign to the queue, the fewer fields they have in common.

The database record fields that are common to all service operations in the queues appear in the queues's Common Field list. If only one service operation is assigned to the queues, all of its fields appear on the partitioning list.

To designate a field as a partitioning key, select the check box next to its name.

Message Header Fields

The message header fields are system-maintained fields that are common to all service operations, regardless of format. If a queue includes any nonrowset-compatible service operations, the message header fields are the only ones that PeopleSoft Integration Broker recognizes as common to every service operation. You can view them as part of the message XML in Integration Broker Monitor. You can also access some of them using equivalent PeopleCode Message class properties, as indicated later in this chapter. The message header fields are:

- OPERATIONNAME: This field contains the name of the operation in the PeopleSoft Pure Internet Architecture.

 Copyright © 1988-2007, Oracle. All rights reserved.

- PUBLISHER: This field contains the user ID that is in effect when the service operation is published, that is, the ID of the user who is signed in to the publishing database.
- PUBPROC: This field refers to the PeopleSoft process that publishes the service operation.

  It is generated when the service operation is published, and it can be the name of a component, an Application Engine program, or an iScript program.

These header fields are always available in the queue's partitioning field list. To designate a field as a partitioning key, select the Include check box next to its name.

Message XML Fields

The message XML fields comprise all the fields that exist anywhere in a message, including PeopleSoft common application message attributes (PSCAMA) record fields. Such fields may not be visible in the queues's partitioning field list, but they are valid for partitioning. Message XML fields can have aliases, allowing for support of mixed-case names.

To designate a message XML field as a partitioning key:

1. On the Queue Definition page, click the Add Field button.
2. Enter the tag name of the XML field, or click the Lookup button to search for one.

   The value does not have to be in the database.

   All names are uppercase by default. You can then add an alias, which can be mixed-case for partitioning.

At runtime, the integration engine searches each message for the first instance of that field tag and uses the value that is associated with it for partitioning. Therefore, if you have common fields in the PSCAMA record that are specific to a batch publish set, you can use those fields.

Renaming and Deleting Queues

You can rename and delete queue definitions using the Queue page in the Service Administration component.

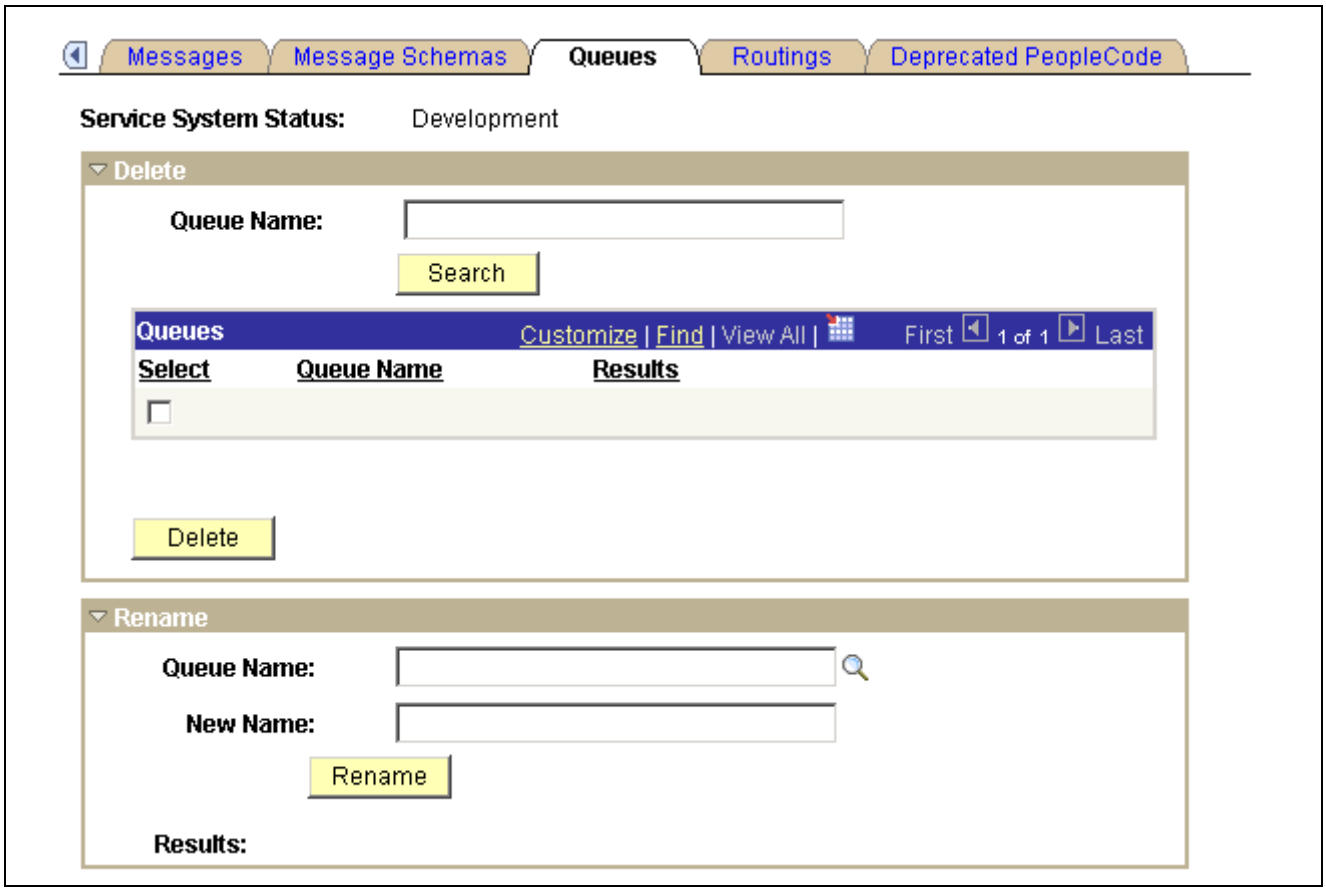
The Queues page contains two sections: a Delete section that enables you to delete a queue definition and a Rename section that enables you to rename a queue definition.

To access the page, select PeopleTools, Integration Broker, Service Utilities, Services Administration and select the Queues tab.

When you first access the page, both sections are collapsed. Click the section-header arrow buttons to expand and collapse each section.

The following example shows the Queues page with the Delete and Rename sections expanded:

Copyright © 1988-2007, Oracle. All rights reserved.



Services Administration - Queues page with the Delete and Rename sections expanded

The top of the page displays a Service System Status field with the current setting, as defined on the Services Configuration page. This setting affects the ability to rename and delete queues.

See Chapter 14, "Managing Services," Configuring PeopleSoft Integration Broker for Handling Services, page 277.

Pages Used to Rename and Delete Queue Definitions

Page Name	Object Name	Navigation	Usage
Service Administration-Queues page	IB_HOME_PAGE3	Select PeopleTools, Integration Broker, Service Utilities, Service Administration. Click the Queues tab.	Rename and delete queue definitions.

Renaming Queue Definitions

To rename a queue definition:

1. Access the Services Administration - Queues page.
2. Click the arrow next to the Rename section header to expand the section.
3. In the Queue Name field, enter the queue definition to rename, or click the Lookup button to search for and select the queue to rename.

 Copyright © 1988-2007, Oracle. All rights reserved.

4. In the New Name field, enter the new name for the queue definition.
5. Click the Rename button.

Deleting Queue Definitions

To delete a message definition:

1. Access the Services Administration - Queues page.
2. Click the arrow next to the Delete section header to expand the section.
3. In the Queue Name field, enter the name of the queue definition to delete, and click the Search button.
   Search results appear in the results grid.
4. In the results grid, select the check box next to the queue or queues to delete.
5. Click the Delete button.

Deleting Queues During Upgrade

To delete a queue definition in an application upgrade project, first ensure that no live instances of messages are assigned to that queue. Archive or delete any such messages in both the source and the target database. Otherwise, an error message appears during the copy process indicating that the message is in use.

Copyright © 1988-2007, Oracle. All rights reserved.

 Copyright © 1988-2007, Oracle. All rights reserved.

CHAPTER 12

Sending and Receiving Messages

This chapter discusses how to:

- Generate and send messages.
- Receive and process messages.
- Process inbound errors.
- Use Message object functionality with nonrowset-based messages.
- Generate test messages.
- Work with message segments.

Understanding Sending and Receiving Messages

To send and receive messages you use PeopleCode to:

- Send request messages from PeopleSoft Integration Broker to other systems.
- Receive response messages from other systems.
- Route messages.
- Manipulate message content.

Note. You can also send messages directly to the integration gateway, thereby bypassing processing on the integration engine.

Prerequisites for Sending and Receiving Messages

Before you can define PeopleCode to generate, send, receive, and process messages, you must define the message in PeopleSoft Internet Architecture.

Note. Once you create PeopleCode, you must also define nodes, services and service operations to implement a complete integration.

Messaging Process Flows

The integration engine uses asynchronous request processes and synchronous request processes to manage outbound and inbound messages. These processes examine the messaging elements that you create to determine how to treat each message.

Copyright © 1988-2007, Oracle. All rights reserved.

Outbound Message Processing Flow

This section discusses message processing flow for outbound messages. In this section, the term process is used, and refers to either the integration engine's asynchronous request process or its synchronous request process, depending on the type of integration you are preforming.

Outbound messages you send go through the following steps.

1. The application triggers the sending PeopleCode that you developed.
2. The PeopleCode program populates and sends the message by using an asynchronous or synchronous method.
3. The method that the PeopleCode uses to send the message triggers a request process in the application's integration engine.
4. The process searches the outbound routings that are associated with that service operation to determine the valid target nodes for the message.

   The asynchronous process examines only asynchronous routings, and the synchronous process examines only synchronous routings. If for synchronous processing, a valid single outbound routing cannot be found, the sending method returns an error.

   Note. Only active routings are considered for processing.

5. For each outbound routing that it finds, the process submits the message to the local gateway, along with transaction information about the node and the target connector that should be used to send the message.
6. The local gateway transmits the message to the specified target node through the specified target connector.
7. If this is a synchronous message, the process waits for the target node to pass a response message back through the gateway, then returns it to the calling PeopleCode method.

Inbound Message Processing Flow

Each received message goes through the following steps:

1. The application's local gateway receives a request message from a remote node or gateway, which specifies the application as its target node.
2. The local gateway submits the message to the application's integration engine, which searches for any inbound request routing parameter which has the same alias name as the external operation name passed in.
3. If a matching routing alias name isn't found, the integration engine returns an error message through the gateway to the sending node.

   If a routing alias name is found, the integration engine invokes either the asynchronous request process or the synchronous request process, as appropriate, to handle the message.

   Note. Any inbound routing alias that is found must have the proper permissions for that service operation for the process to proceed.

4. The process accesses the service operation that matches the routing alias name and passes the message to the service operation's handler associated with receiving PeopleCode.

 Copyright © 1988-2007, Oracle. All rights reserved.

- The asynchronous request process invokes the service operation's handler OnNotify event PeopleCode.
- The synchronous request process invokes the service operation's handler OnRequest event PeopleCode.

5. If this is a synchronous transaction, the process waits for the receiving PeopleCode to generate and return a response message, then passes it back to the sending node through the gateway.

Understanding Integration PeopleCode

This section discusses the PeopleCode used for integrations and describes:

- Sending and receiving PeopleCode.
- Integration application classes.
- Integration methods.
- Messaging methods.
- Error-handling methods.
- Messaging PeopleCode.

Sending and Receiving PeopleCode

This section discusses the PeopleCode you use for sending messages from PeopleSoft Integration Broker to other systems, and the PeopleCode you use for receiving messages from other systems.

Sending PeopleCode

PeopleCode for sending messages can be located in PeopleCode events associated with records, record fields, and components, and in application engine programs.

The PeopleCode method used to send messages is highlighted in the following table.

Transmission Type	Sending PeopleCode	Comments
Synchronous	SyncRequest method.	The SyncRequest method belongs to the IntBroker class.
Asynchronous	Publish method.	The Publish method belongs to the IntBroker class.

To work with messages in SOAP format, transform the SOAP documents into XML documents and then use the IntBroker class SyncRequest or Publish methods.

Receiving PeopleCode

The PeopleCode that you use to receive a message must be associated with the message definition. The transmission type of the message determines the location of the PeopleCode program.

Implement the OnRequest method for synchronous messages. Implement the OnNotify method for asynchronous messages. Both methods are located in the PS_PT application package, in the Integration sub-package, in the IRequestHandler and INotificationHandler classes, respectively.

Copyright © 1988-2007, Oracle. All rights reserved.

Transmission Type	Message Structure	Receiving PeopleCode	Comments
Synchronous	Rowset-based	Message is passed into the method.	Implement the OnRequest method in the IRequestHandler application interface.
Synchronous	Nonrowset-based	Message is passed into the method.	Implement the OnRequest method in the IRequestHandler application interface.
Asynchronous	Rowset-based	Message is passed into the method.	Implement the OnNotify method in the INotificationHandler application interface.
Asynchronous	Nonrowset-based	Message is passed into the method.	Implement the OnNotify method in the INotificationHandler application interface.

To get content data out of a message, use the following guidelines.

Message Structure	PeopleCode	Comments
Rowset-based	GetRowSet method.	None.
Nonrowset-based	GetXMLDoc method.	You can also use Message class functionality with nonrowset-based messages. See Chapter 12, "Sending and Receiving Messages," Using Message Object Functionality With Nonrowset-Based Messages, page 253.

Application Classes

Application classes house the processing logic for asynchronous and synchronous messages. By implementing the Integration Broker application classes, you can reuse code and access other benefits of application classes.

The following application classes exist for PeopleSoft Integration Broker. To access these application classes, in PeopleSoft Application Designer, open the PS_PT application package and open the Integration subpackage.

Note. All of the Integration Broker application classes are defined as interfaces. This means that there is no native implementation of them: you must import them to your program and implement them if you want to use them.

Application Class	Methods Contained in Application Class	Comments
INotificationHandler	• OnNotify • OnError	This interface is the equivalent of the Subscription Message event PeopleTools releases prior to PeopleTools 8.48.
IReceiver	• OnAckReceive • OnError	This interface is the equivalent of the OnAckReceive Message event in PeopleTools releases prior to PeopleTools 8.48.

 Copyright © 1988-2007, Oracle. All rights reserved.

Application Class	Methods Contained in Application Class	Comments
IRequestHandler	• OnRequest • OnError	This interface is the equivalent of the OnRequest Message event in PeopleTools releases prior to PeopleTools 8.48.
IRouter	• OnRouteSend • OnRouteReceive • OnError	This interface is the equivalent of the OnRouteSend and OnRouteReceive Message events in PeopleTools releases prior to PeopleTools 8.48.
ISend	• OnRequestSend • OnError	This interface is the equivalent of the OnSend Message event in PeopleTools releases prior to PeopleTools 8.48.

Each of the methods contained in these application classes is described in this section.

Routing Methods

Routing methods determine how a message is routed to or from PeopleSoft Integration Broker.

OnRouteSend Method

Implement the OnRouteSend method for outbound synchronous and asynchronous service operations to specify to what node PeopleSoft Integration Broker routes a message. The implementation of this method enables you to apply PeopleCode that filters the destination nodes to which PeopleSoft Integration Broker routes messages.

The OnRouteSend method is contained in the IRouter application class, which is contained in the PS_PT application package, in the Integration subpackage.

When the application PeopleCode is invoked to send a message, the transaction definitions in the local database provide a list of target nodes to which PeopleSoft Integration Broker can route the message. The integration engine's request handler invokes the service operation's OnRouteSend event. You can implement the OnRouteSend method in the application package associated with the handler for this service operation, which enables you to apply additional PeopleCode that determines the final target nodes.

You can use OnRouteSend to validate the outbound service operation's target node list, prevent the message from transmitting, or redirect it to a completely different set of targets.

OnRouteSend enables you to account for multiple synchronous targets. Only one target node at a time can receive a request message sent with a synchronous transaction. Even though you can define the same outbound synchronous transaction for multiple nodes, you must make sure the transaction resolves to a single target node or the transaction fails.

The following is an implementation of this class:

```
import PS_PT:Integration:IRouter;

class RoutingHandler implements PS_PT:Integration:IRouter
   method RoutingHandler();
   property array of any destinationNodes;
   method OnRouteSend(&_MSG As Message) Returns integer;
end-class;

/* constructor */
```

Copyright © 1988-2007, Oracle. All rights reserved.

```
method RoutingHandler
end-method;

method OnRouteSend
   /+ &_MSG as Message +/
   /+ Returns Integer +/
   /+ Extends/implements PS_PT:Integration:IRouter.OnRouteSend +/
   /* Variable Declaration */
   Local any &aNodeList;
   Local any &rootNode;
   Local any &xmlDoc;

   /*
* Check the message for the instructions on how to execute
* the OnRouteSend.
*
*/

   &xmlDoc = &_MSG.GetXmlDoc();
   &rootNode = &xmlDoc.DocumentElement;
   &aNodeList = &rootNode.GetElementsByTagName("OnRouteSend");

   If (&aNodeList.Len <> 1) Then

      /* A single node must be present. */
      Exit;
   Else

      /* check the value of the node to determine the action to
       take. */

      Evaluate &aNodeList [1].NodeValue
      When "True"
         Return (%IntBroker_ROUTE_ALL);
         Break;
      When "False"
         Return (%IntBroker_ROUTE_NONE);
         Break;
      When-Other

         /* assume that this is to be routed to the node given */
         Local array &nodeArray;
         &nodeArray = CreateArray();
         &nodeArray.Push(&aNodeList [1].NodeValue);

         Local string &sIBVariableTest = GetCurrentType(&nodeArray);
         Evaluate &sIBVariableTest
         When "Array"
            &destinationNodes = &nodeArray.Clone();
            Return %IntBroker_ROUTE_SOME;
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
            When "BooleanTrue"
               Return %IntBroker_ROUTE_ALL;
            When "BooleanFalse"
               Return %IntBroker_ROUTE_NONE;
            End-Evaluate;
            Break;

         End-Evaluate;

      End-If;

   end-method;
```

OnRouteReceive Method

Implement the OnRouteReceive method for inbound synchronous and asynchronous service operations to apply PeopleCode that determines whether the default local node accepts inbound messages.

The OnRouteReceive method is contained in the IRouter application class, which is contained in the PS_PT application package, in the Integration subpackage.

When the integration engine receives a message, the transaction definitions in the local database provide a list of source nodes from which the application can accept the message. The integration engine's request handler invokes the service operation's OnRouteReceive event. You can implement the OnRouteReceive method in the application package associated with the handler for this service operation, which enables you to apply PeopleCode that determines whether the default local node accepts the inbound message. You can employ this event regardless of the message transmission type.

The following is an example implementation of this method:

```
import PS_PT:Integration:IRouter;

class RoutingHandler implements PS_PT:Integration:IRouter
   method RoutingHandler();
   property array of any destinationNodes;
   method OnRouteReceive(&_MSG As Message) Returns boolean;
end-class;

/* constructor */
method RoutingHandler
end-method;

method OnRouteReceive
   /+ &_MSG as Message +/
   /+ Returns Boolean +/
   /+ Extends/implements PS_PT:Integration:IRouter.OnRouteReceive +/
   /* Variable Declaration */
   Local any &aNodeList;
   Local any &rootNode;
   Local any &xmlDoc;
```

Copyright © 1988-2007, Oracle. All rights reserved.

```
   /*
*  Check the message for instructions on how to execute
*  the OnRouteReceive.
*
*/

   &xmlDoc = &_MSG.GetXmlDoc();
   &rootNode = &xmlDoc.DocumentElement;
   &aNodeList = &rootNode.GetElementsByTagName("OnRouteReceive");

   If (&aNodeList.Len <> 1) Then

      /* A single node must be present. */
      Exit;
   Else

      /* check the value of the node to determine the action to
        take. */

      Evaluate &aNodeList [1].NodeValue
      When "True"
         Return ( True);
         Break;
      When "False"
         Return ( False);
         Break;
      When-Other
         /* don't recognize the value. */
         Exit;
      End-Evaluate;

   End-If;

end-method;
```

Messaging Methods

This section describes methods used in messaging and the application classes in which they are contained

Outbound Messaging Methods

This section describes methods used on outbound messages from PeopleSoft to other systems.

OnRequestSend Implement for outbound synchronous and asynchronous service operations to override connector properties before sending a message to the integration gateway.

This method is contained in the ISend application class.

 Copyright © 1988-2007, Oracle. All rights reserved.

The OnRequestSend method passes in a message to your derived application class method. The return needs to be a message.

The following is an example implementation of this method.

```
import PS_PT:Integration:ISend;

class SendHandler implements PS_PT:Integration:ISend
   method SendHandler();
   method OnRequestSend(&_MSG As Message)
    Returns Message;
end-class;

/* constructor */
method SendHandler
end-method;

method OnRequestSend
   /+ &_MSG as Message +/
   /+ Returns Message +/
   /+ Extends/implements PS_PT:Integration:ISend. +/
   /+ OnRequest Send +/
   /* Variable Declaration */
   Local any &tempNode;
   Local any &rootNode;
   Local any &xmlDoc;
   Local any &msg;

   &msg = &_MSG;

   &xmlDoc = &msg.GetXmlDoc();

   /* Add a node to the doc to prove that we can
      edit it in this event. */

   &rootNode = &xmlDoc.DocumentElement;

   &tempNode = &rootNode.AddElement("OnSend");
   &tempNode.NodeValue = "If you see this, then
   the Sync OnSend PCode has altered the message";

   /* and write the data back into the message */

   &msg.SetXmlDoc(&xmlDoc);

   Return (&msg);

end-method;
```

Copyright © 1988-2007, Oracle. All rights reserved.

See Chapter 12, "Sending and Receiving Messages," Setting and Overriding Target Connector Properties at Runtime, page 228.

When using the ISendHandler with message parts, specifically with rowset-based message parts, the rowsets of the parts must be retrieved *in the order* that the content data will be sent .

The following is an example that can be used for ISend events that use rowset-based parts (even for the cases where one is just overriding the connectors):

```
method OnRequestSend
   /+ &MSG as Message +/
   /+ Returns Message +/
   /+ Extends/implements PS_PT:Integration:ISend. +/
   /+ OnRequestSend +/
   If (&MSG.IsPartsStructured) Then
      Local number &i;
      Local Rowset &rs;
      For &i = 1 To &MSG.PartCount
         &rs = &MSG.GetPartRowset(&i);
      End-For;
   End-If;

      Return &MSG;

end-method;
```

OnAckReceive

Implement for outbound asynchronous service operations to access the body of a message acknowledgement to check for SOAP faults.

This method is contained in the IReceiver application class.

The following is an example implementation of this method.

```
import PS_PT:Integration:IReceiver;

class AckReceiveHandler implements PS_PT:
  Integration:⇒
IReceiver
   method AckReceiveHandler();
   method OnAckReceive(&_MSG As Message) Returns
   integer;
end-class;

/* constructor */
method AckReceiveHandler
end-method;

method OnAckReceive
   /+ &_MSG as Message +/
   /+ Returns Integer +/
   /+ Extends/implements PS_PT:Integration:+/
   /+ IReceiver.OnAck Receive +/
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
   /* Variable Declaration */
   /*
/* We return a hardcoded value. In this case, a
   message error.*/

   Return (%Operation_Error);

end-method;
```

See Chapter 12, "Sending and Receiving Messages," Handling Inbound Asynchronous Transactions, page 232.

Inbound Messaging Methods

This section describes methods used on inbound messages to PeopleSoft from other systems.

OnRequest

Implement for inbound synchronous service operations when a response is required.

This method is contained in the IRequestHandler application class.

The following is an example implementation of this method:

```
import PS_PT:Integration:IRequestHandler;

class RequestHandler implements PS_PT:Integration:
 IRequest Handler
   method RequestHandler();
   method OnRequest(&_MSG As Message) Returns
   Message;
end-class;

/* constructor */
method RequestHandler
end-method;

method OnRequest
   /+ &_MSG as Message +/
   /+ Returns Message +/
   /+ Extends/implements PS_PT:Integration:+/
   /+ IRequest Handler.OnRequest +/
   /* Variable Declaration */
   Local any &tempNode;
   Local any &descNode;
   Local any &rootNode;
   Local any &xmlDoc;
   Local any &xmldata;
   Local any &msg;

   &msg = CreateMessage(Operation.QE_IB_SYNC_RESP);
```

Copyright © 1988-2007, Oracle. All rights reserved.

```
   &xmldata = "<?xml version='1.0'?>
   <QE_IB_PeopleCode_Test⇒/>";

   &xmlDoc = CreateXmlDoc(&xmldata);

   &rootNode = &xmlDoc.documentelement;

   &descNode = &rootNode.AddElement("Description");
   &descNode.NodeValue = "Sync test of OnRouteSend.";

   &tempNode = &rootNode.addelement("OnRequest");
   &tempNode.NodeValue = "If you see this,
   then the On Request PCode created the response
   message";

   &msg.SetXmlDoc(&xmlDoc);

   Return &msg;


end-method;
```

OnNotify

Implement for inbound asynchronous service operations. This method can be used for code that does subscription processing, and for validating and loading message data.

This method is contained in the INotificationHandler application class.

The following is an example implementation of this method:

```
import PS_PT:Integration:INotificationHandler;

class RESPONSE_NOTIFICATION implements PS_PT:
 Integration:INotificationHandler
   method RESPONSE_NOTIFICATION();
   method OnNotify(&MSG As Message);

end-class;

/* constructor */
method RESPONSE_NOTIFICATION
   %Super = create PS_PT:Integration:INotification
   Handler⇒
();
end-method;

method OnNotify
   /+ &MSG as Message +/
   /+ Extends/implements PS_PT:Integration:+/
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
      /+ INotification Handler.OnNotify +/

      Local Rowset &rs;
      Local boolean &Ret;
      Local string &TransactionID;

      &rs = &MSG.GetRowset();

      If &MSG.IsSourceNodeExternal Then

         /* if the request came from an external non
           PeopleSoft System then you can get the
           original TransactionID from the WSA_MessageID
           from the request message. */

         &TransactionID = &MSG.IBInfo.WSA_MessageID;

      Else

         /* if the request came from a PeopleSoft
            System then get the original TransactionID
            from the  nReplyToID */

         &TransactionID = &MSG.IBInfo.InReplyToID;

      End-If;

   end-method;
```

Error-Handling Methods

Each application class contained in the Integration application subpackage contains an OnError method that you can use for custom error handling.

Custom error handling can include sending an email notification or entering data in a log when an error occurs.

For the IRequestHandler application class, the OnError function returns a string. This enables you to send back custom error messages, for example SOAP faults, to non-PeopleSoft consumers. If the message consumed was a SOAP message and the custom error message is already wrapped in SOAP, it will not be modified and is sent as-is. However, if the OnError message is not SOAP, it is wrapped as a standard SOAP fault and returned to the sender.

If the message consumer is another PeopleSoft system the message set/message ID framework applies.

If an error occurs the OnError method, if implemented, is automatically invoked. The type of exception can be viewed by using the Message object to retrieve an Exception object populated with information about the error, using the message class IBException property.

The following is an example of the OnError method implementation:

```
/*On Error Implementation */
```

Copyright © 1988-2007, Oracle. All rights reserved.

```
method OnError
   /+ &MSG as Message +/
   /+ Returns String +/
   /+ Extends/implements PS_PT:Integration:IRequestHandler.OnError +/
   Local integer &nMsgNumber, &nMsgSetNumber;
   Local string &sText;

   &nMsgNumber = &MSG.IBException.MessageNumber;
   &nMsgSetNumber = &MSG.IBException.MessageSetNumber;
   rem    &sText = &exception.DefaultText;
   &sText = &MSG.IBException.ToString();


   /* ADD SPECIFIC ERROR INFO HERE */
   Return &sText;

end-method;
```

See *Enterprise PeopleTools 8.49 PeopleBook: PeopleCode API Reference*, “Exception Class”.

See *Enterprise PeopleTools 8.49 PeopleBook: PeopleCode API Reference*, “Message Classes,” IBException.

Messaging PeopleCode

Messaging PeopleCode enables you to manipulate message content. The messaging PeopleCode classes you can use for this are:

Message classes	Use for rowset or nonrowset-based messages.
SOAPDoc class	Use for SOAP-compliant messages.
XMLDoc classes	Use for XML DOM-compliant messages.

XML DOM-compliant and SOAP-compliant messages are nonrowset-based messages. You can use their respective classes to manipulate message content, or use the Message classes.

See Also

Chapter 12, “Sending and Receiving Messages,” Using Message Object Functionality With Nonrowset-Based Messages, page 253

Enterprise PeopleTools 8.49 PeopleBook: PeopleCode API Reference, “Message Classes”

Enterprise PeopleTools 8.49 PeopleBook: PeopleCode API Reference, “SOAPDoc Class”

Enterprise PeopleTools 8.49 PeopleBook: PeopleCode API Reference, “XmlDoc Class”

Messaging Handlers

Messaging handlers, or *handlers* are associated with a service operation on the Handlers tab of the Service Operations page.

 Copyright © 1988-2007, Oracle. All rights reserved.

Handlers define additional programming to be used with processing the message associated with the service operation.

The following are the different types of handlers:

- OnNotify
- On Receive
- On Request
- On Response
- On Route
- On Send

See Also

Chapter 15, "Managing Service Operations," Adding Handlers to Service Operations, page 301

Selecting Handlers

The availability of each handler type depends on the type of service operation you are using. The following table lists the message structures used for each service operation type and the handler types available for use.

Service Operation Type	On Notify Handler Type	On Receive Handler Type	On Request Handler Type	On Response Handler Type	On Route Handler Type	On Send Handler Type
Asynchronous one-way	Request message	Request message	NA	NA	Request message	Request message
Asynchronous request /response	Request message	Request message	NA	Response message	*Request message *Response message	Request message
Asynchronous to synchronous	Request message	NA	Request message	Response message	*Request message *Response message	Request message
Synchronous	NA	NA	Request message	NA	Request message	Request message

Note. * For On Route with On Send, the message structure is a request message. For On Route with On Receive, the message structure is a response message.

The On Response handler type is used to identify the type of OnNotify event to be fired. For example, assume there are four OnNotify handlers that are to be fired—three are general OnNotify events, that is, the message is processed as part of the request, and the fourth is a response to the original asynchronous request. The fourth one is specified with a handler type of On Response, and the application class selected is the base class OnNotify.

Copyright © 1988-2007, Oracle. All rights reserved.

Implementing Handlers

You can implement handlers using application classes, component interfaces, data mover scripts or pre-PeopleTools 8.48 integration PeopleCode constructs.

The following table lists the handlers you can implement using application classes, component interfaces and data mover scripts:

Implementation	On Notify Handler	On Receive Handler	On Route Handler	On Send Handler	On Request Handler	On Response Handler
Application class	Y	Y	Y	Y	Y	Y
Component interface	Y	N	N	N	Y	Y
Data mover script	Y	N	N	N	N	N

Implementing Handlers Using Application Classes

You can specify an application class as a handler for a service operation. This is the most typical implementation of a handler.

For application class handlers, the names that populate the drop down used for selecting the appropriate method must have the exact signature required for the method.

Handler Type	Input Parameter to Method	Method Output Parameter
OnNotify	Message	Void (none)
OnResponse	Message	Void (none)
OnReceive	message	Int
OnRequest	Message	Message
OnSent	Message	Message
OnRoute	Message	Integer or Boolean

Note. For OnRoute: if you select a method that returns as integer, the handler type is OnRouteSend. If you select a method that returns as Boolean, the handler type is OnRouteReceive.

To implement a handler using an application class:

1. Select the Integration Broker method that you want to implement based on the type of service operation you are creating.
2. Create a new application class, and import the appropriate Integration Broker application class. For example:

```
import PS_PT:Integration:INotificationHandler;
```


Copyright © 1988-2007, Oracle. All rights reserved.

3. Define a class that implements the Integration Broker application class.
4. Define the method that implements the Integration Broker method, with the appropriate signature. In the following example, the OnNotify method would be available as a handler method.

```
class RESPONSE_NOTIFICATION implements PS_PT:Integration:INotificationHandler

method RESPONSE_NOTIFICATION();
method OnNotify(&MSG As Message);

end-class;
```

5. In the definition of the class, create the program-specific code to be used for this handler.
6. When you define the service operation, select the application package, class and method you created as part of the handler details.

Implementing Handlers Using Component Interfaces

You can implement a component interface as a handler to access extant business rules and processes to be used with the message data.

You can only use a component interface for On Notify, On Request and On Response messaging handler types.

Component interfaces can be used as handlers for both asynchronous as well as synchronous messages. Asynchronous messages should only be used with methods that do not require a response, such as Update or Insert.

You must define a component interface as a service before you can use implement any of the methods as handlers.

The standard methods for a component interface (such as Find, Get, Save, Update, and so on) are automatically available for a handler. However, if you want to use a user-defined method, you must include the keyword **Doc** as part of the signature.

See Chapter 17, “Creating Component Interface-Based Services,” page 321.

Implementing Handlers Using Data Mover Scripts

You can implement a handler using a data mover script (DMS) to bulk load large amounts of data into a local table. DMS can only be selected for asynchronous one way service operations, and can only be used with rowset-based request messages.

Use DMS as a handler type for large messages.

When the OnNotify event is fired, the message data is inserted into the tables defined in the message using data mover script constructs.

Data Mover doesn’t perform any data validations.

Generating and Sending Messages

This section provides an overview of outbound messaging and discusses how to handle:

- Outbound asynchronous message transmission.

Copyright © 1988-2007, Oracle. All rights reserved.

- Outbound synchronous message transmission.
- Cookies in messages.

Understanding Outbound Messaging

Successful outbound messaging relies on sending messages in the proper order and on testing the messages. Messages containing non-XML data have special considerations.

Message Order

PeopleSoft Integration Broker guarantees that messages are delivered in the order in which you send them and that they are single-threaded at the PeopleSoft receiving node. However, message order is not part of the queue definition. You must send messages in the proper order.

Note. You can modify this behavior by using queue partitioning.

See Chapter 11, "Managing Service Operation Queues," Applying Queue Partitioning, page 199.

Message Testing

Make sure that you adequately unit-test and system-test your messages.

Unit-test a message by triggering the PeopleCode that sends the message and then view the message details in Service Operations Monitor. From the Service Operations Monitor, you can view the contents of each field to verify that the message data is formatted correctly.

See Chapter 21, "Using the Service Operations Monitor," page 413.

You can also test handler code using the Handler Tester utility.

See *Enterprise PeopleTools 8.49 PeopleBook: Integration Testing Utilities and Tools*

Message Class Outbound PeopleCode

Use the record class SelectByKey method whenever possible to get data that isn't in the component buffer.

If the record names are the same, use the standard record methods, such as SelectByKey, Insert, and Update, on the message records.

There are no automatic checks for message size. You must do it programatically. Use the following code at level 0 to control message size when dealing with multiple transactions:

```
If &Msg.Size > %MaxMessageSize
```

Note. The OnRouteSend method enables you to apply PeopleCode that filters the destination nodes.

See *Enterprise PeopleTools 8.49 PeopleBook: PeopleCode API Reference*, "Record Class".

Non-XML Data

If you're generating a non-XML outbound message, it's up to you to insert the message content into a special XML section containing a CDATA tag:

```
<xml psnonxml="yes">
  <![CDATA[nonXML_message_data]]>
```

 Copyright © 1988-2007, Oracle. All rights reserved.

Handling Outbound Asynchronous Message Transmission

. To send a message asynchronously, use the IntBroker class Publish method in:

- A record field PeopleCode event.
- A component PeopleCode event.

  When publishing from a component, publish messages only from the SavePostChange event, using the deferred mode property.
- An Application Engine program.
- An implementation of the OnNotify method.
- An implementation of the OnRequest method .

  The OnRequest service operation event is triggered only when an inbound transaction occurs. However, when the receiving PeopleCode runs, the program can also send messages.

Message Class Outbound Asynchronous Example

The following example uses the Publish method in the PeopleCode IntBrokerclass:

```
Local Message &MSG;
Local Rowset &SALES_ORDER, &RS;

 /*Get a pointer to the component buffer rowset */
&SALES_ORDER = GetLevel0();
/*Create an instance of the SALES_ORDER_ASYNC message object */
&MSG = CreateMessage(OPERATION.SALES_ORDER_ASYNC);

/*Copy the rows from the rowset to the message object */
&MSG.CopyRowset(&SALES_ORDER);
/*Send the message */
%IntBroker.Publish(&MSG);
```

XmlDoc Class Outbound Asynchronous Example

The following example uses the Publish method:

```
Local XmlDoc &xmlRequestDoc;
Local Message &MSG;

/*Create an XmlDoc Object */
&xmlRequestDoc = CreateXmlDoc();

/* Parse an input XML file into an XmlDoc */

&bool = &xmlRequestDoc.ParseXmlFrom URL("C:\pt\appserv\files\
input.xml");

/* Populate message with XML data */
&MSG = CreateMessage(OPERATION.XmlRequest);

&MSG.SetXmlDoc(&xmlRequestDoc);
```

Copyright © 1988-2007, Oracle. All rights reserved.

```
/* Sent request */

%IntBroker.Publish(&MSG);
```

Identifying SOAP Faults

You can implement the OnAckReceive method to access IBInfo data. This enables you to read the content of acknowledgements returned by recipient systems of asynchronous SOAP messages. The ability to access acknowledgement content is useful when sending SOAP messages, since although there may be no HTTP protocol errors while sending them, SOAP faults may occur.

If the message is nonrowset-based, use the message class GetXmlDoc method to get the response data. This returns an XmlDoc object.

If the message is rowset-based, use the message class GetXMLString method to get the response data. This returns a string object which you can load into an XmlDoc object.

If SOAP faults are found, you can set the status equal to Error so that the errors appear in Integration Broker Monitor for the Publication Contract.

The following code example shows how to use GetXmlDoc and GetXMLString in an implementation of the OnAckReceive method. Valid status overrides are %Operation_Done, %Operation_Error, and %Operation_Retry.

```
import PS_PT:Integration:IReceiver;

class AckReceiveHandler implements PS_PT:Integration:IReceiver
   method AckReceiveHandler();
   method OnAckReceive(&_MSG As Message) Returns integer;
end-class;

/* constructor */
method AckReceiveHandler
end-method;

method OnAckReceive
   /+ &_MSG as Message +/
   /+ Returns Integer +/
   /+ Extends/implements PS_PT:Integration:IReceiver.OnAckReceive +/
   /* Variable Declaration */
   /*
*/

If &MSG.IsStructure Then

/* if message is rowset-based */
&str = &MSG.GenXMLString();

Else
/* if message is nonrowset-based */
&xmldoc = &MSG.GetXmlDoc();
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
End-If;

 Return (%Operation_Done);

end-method;
```

You can also implement the OnAckReceive method to read response content data returned from third-party systems when using the HTTP target connector.

See Also

Enterprise PeopleTools 8.49 PeopleBook: PeopleCode API Reference, "XmlDoc Class"

Handling Outbound Synchronous Transactions

Use the IntBroker class SyncRequest method for handling outbound synchronous transfers. To send a message synchronously, you can employ SyncRequest in:

- The record field SavePreChange PeopleCode event.
- The record field SavePostChange PeopleCode event.
- The record field Workflow PeopleCode event.
- The record field FieldChange PeopleCode event.
- An implementation of the OnRequest method.
- An implementation of the OnNotify method.

Note. The OnRequest and OnNotify events are triggered only when an inbound transaction occurs, however, when the receiving PeopleCode runs, it can also send messages.

The response message that is returned from an outbound synchronous transaction is no different from an inbound request message. Once you have it in a Message, XmlDoc, or SoapDoc object, it has the same properties as any other object of that type and can, therefore, be treated exactly the same way.

See Chapter 12, "Sending and Receiving Messages," Receiving and Processing Messages, page 232.

Message Class Outbound Synchronous Example 1

The following example uses the IntBroker class SyncRequest method:

```
Local Message &MSG, &response;
Local Rowset &SALES_ORDER;

&SALES_ORDER = GetLevel0();
&MSG = CreateMessage(OPERATION.SALES_ORDER_SYNC);
&MSG.CopyRowsetDelta(&SALES_ORDER);

/* send the synchronous request; the return value is the response
message object */
&response = %IntBroker.SyncRequest(&MSG);

/* check the response status; 0 means OK */
```

Copyright © 1988-2007, Oracle. All rights reserved.

```
If (&response.ResponseStatus = 0) Then
  /* process the response */
  MY_SALES_ORDER_SYNC.ORDER_ID = &response.GetRowset().GetRow(1)
.GetRecord(Record.SO_RESPONSE).GetField(Field.ORDER_ID).Value);
else

  /* do error handling */

End-If;
```

Message Class Outbound Synchronous Example 2

The following example shows the use of CopyTo to get the data back from the response and into the component buffer, and therefore the page:

```
Local Message &msgZipRequest, &msgZipResponse;
Local Rowset &RS, &rsMessageRowset;


&RS = GetLevel0();
&msgZipRequest = CreateMessage(OPERATION.ZIP_REQUEST);
&msgZipRequest.CopyRowset(&RS);
/* send the synchronous request; the return value is the response
message object ⇒
 */
&msgZipResponse = %IntBroker.SyncRequest(&msgZipRequest,
Node.ZIPTOCITYANDSTATE);

/* check the response status; 0 means OK */
If (&msgZipResponse.ResponseStatus = 0) Then
   /* process the response */
   &rsMessageRowset = &msgZipResponse.GetRowset();
   &rsMessageRowset.CopyTo(&RS);
else
   /* do error handling */
End-If;
```

XmlDoc Class Outbound Synchronous Example

The following example uses the IntBroker class SyncRequest method:

```
Local string &xmlString;
Local XmlDoc &xmlRequestDoc;
Local XmlDoc &xmlResponseDoc;
Local Field &DescrLong;

&DescrLong = GetLevel0().GetRow(1).GetRecord(Record.QE_FUNCLIB_IB).
DESCRLONG;

/* Get the input XML string */
&xmlString = &DescrLong.Value;
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
/* Parse the input XML string into an XmlDoc */
 &MSG = CreateMessage(OPERATION.XMLSYNCREQ);

/* Send request */
&ResponseMsg = %IntBroker.SyncRequest(&MSG);
&xmlResponseDoc = &ResponseMSG.GetXmlDoc();

/* Display the output XML string */
&DescrLong.Value = &xmlResponseDoc.GenXmlString();
```

See Also

Enterprise PeopleTools 8.49 PeopleBook: PeopleCode API Reference, "XmlDoc Class"

Overriding Synchronous Timeout Intervals at Runtime

For long-running outbound synchronous transactions, you can override the default timeout period the transaction at runtime using the SyncServiceTimeout property. The default synchronous timeout period is five minutes.

The HTTP header file is modified to take this parameter. The value you set is sent to the integration gateway where it is used for the HTTP timeout.

The SyncServiceTimeout property takes a time (in seconds). The property is read-write.

The following code example shows how to use the property. To use this property, note that you must override and setup the target connector properties for the transaction. As the example demonstrates, there are helper methods that load properties based on node or transaction.

```
&MSG.SetXmlDoc((&xmlReq);
&MSG.IBInfo.LoadConnectorPropFromNode(Node.EAI)
&MSG.IBInfo.SyncServiceTimeout(360000);
&MSG.IBInfo.ConnectorOverride = true;
&MSG_Resp = %IntBroker.SyncRequest(&MSG, Node.EAI);
&xmlResponseDoc = &MSG.GetXmlDoc();
```

See Also

Chapter 12, "Sending and Receiving Messages," Setting and Overriding Target Connector Properties at Runtime, page 228

Enterprise PeopleTools 8.49 PeopleBook: System and Server Administration, "PeopleSoft Timeout Settings"

Handling Cookies

PeopleSoft Integration Broker provides basic cookie handling for exchanges that are initiated by your PeopleSoft application. You can accept a synchronous response message containing cookies, save those cookies in a global variable, and later return them to the remote node in an outbound synchronous or asynchronous request message. This is a typical application of cookies in a web interaction.

Cookies are implemented as an IBInfo class property, Cookies. You can access this property only in an inbound synchronous response message or an outbound request message.

Copyright © 1988-2007, Oracle. All rights reserved.

Receiving Cookies Example

The following example retains the cookies from a response message to a global variable:

```
Local Message &SalesRequest, &SalesResponse;
Local Rowset &SALES_ORDER;
Global string &SalesCookies;

&SALES_ORDER = GetLevel0();
&SalesRequest = CreateMessage(OPERATION.SALES_ORDER_SYNC);
&SalesRequest.CopyRowsetDelta(&SALES_ORDER);

/* Send the synchronous request; the return value is the response
message object */
&SalesResponse = &SalesRequest.SyncRequest();

/* Retrieve cookies from the response message */
&SalesCookies = &SalesResponse.IBInfo.Cookies;
```

Returning Cookies Example

The following example retrieves the previously retained cookies from the global variable and inserts them into a new request message:

```
Local Message &SalesRequest, &SalesResponse;
Local Rowset &SALES_ORDER;
Global string &SalesCookies;

&SALES_ORDER = GetLevel0();
&SalesRequest = CreateMessage(Message.SALES_ORDER_SYNC);
&SalesRequest.CopyRowsetDelta(&SALES_ORDER);

/* Insert the cookies in the request message */
&SalesRequest.IBInfo.Cookies = &SalesCookies;

/* Send the asynchronous request */
%IntBroker.Publish(&SalesRequest);
```

Setting and Overriding Target Connector Properties at Runtime

PeopleSoft Integration Broker enables you to dynamically override target connector properties at runtime that have previously been set at the node, connector and transaction levels. To set or override target connectors at runtime, use the PeopleCode IBInfo object, the Connector Info object and implement the OnRequestSend method.

Note. Properties set at the PeopleCode level take precedence over those set at the node, connector and transaction level.

IBInfo object An IBInfo object is instantiated from a message object.

 Copyright © 1988-2007, Oracle. All rights reserved.

	You can use this object in publishing or synchronous request PeopleCode. You can also use it in your implementation of the OnRequestSend method.
ConnectorInfo object	A ConnectorInfo object is instantiated from an IBInfo object. Use this object for reading and writing connector name/value pair information to and from the IBRequest. You can use this object in publishing or synchronous request PeopleCode. You can also use it in your implementation of the OnRequestSend method.
OnRequestSend Method	The OnRequestSend method is included in the ISend application class. Use your implementation of this method to override target connector properties at runtime for a subscribing node transaction. This event associated with the service operation executes before any transformations are processed. You can use this event for asynchronous and synchronous messages.

Since data is always carried with the message, you can use the IBInfo object, ConnectorInfo object and your implementation of the OnRequestSend method to populate connector information in the publishing PeopleCode and then override it for a specific node.

Setting and Overriding Target Connector Properties Using the OnRequestSend Event

You can use implement the OnRequestSend method to override IBInfo and connector properties at runtime for a subscribing node transaction.

Any content data that is changed on the message or XMLDoc is sent to the subscribing node or used within a transformation.

To override the properties of a target connector, you must set the following statement to *true*:

```
&MSG.IBInfo.ConnectorOverride=true
```

If a publication contract fails as a result of using an implementation of the OnRequestSend method to override connector properties at runtime, correct the PeopleCode in your implementation and resubmit the message.

Example: Setting Target Connector Properties Using the OnRequestSend Method

The following example shows loading all connectors that exist for the node and adding one additional property, Filename.

```
import PS_PT:Integration:ISend;

class SendHandler implements PS_PT:Integration:ISend
   method SendHandler();
   method OnRequestSend(&Msg As Message) Returns Message;
end-class;

/* constructor */
method SendHandler
end-method;

method OnRequestSend
   /+ &MSG as Message +/
   /+ Returns Message +/
```

Copyright © 1988-2007, Oracle. All rights reserved.

```
   /+ Extends/implements PS_PT:Integration:ISend.OnRequestSend +/
   /* Variable Declaration */
   Local Any &Bo;
   Local Message &Msg;

&Bo = &MSG.IBInfo.LoadConnectorPropFromNode("nodename");

&Bo = &MSG.IBInfo.IBConnectorInfo.AddConnectorProperties
      ("FILENAME", "temp", %Property);
&MSG.IBInfo.ConnectorOverride = True;

Return (&Msg);

end-method;
```

Example: Overriding Connector Properties Using the OnRequestSend Method

The following example demonstrates overriding target connector properties using an implementation of the OnRequestSend method for a rowset-based asynchronous message.

```
import PS_PT:Integration:ISend;

class SendHandler implements PS_PT:Integration:ISend
   method SendHandler();
   method OnRequestSend(&Msg As Message) Returns Message;
end-class;

/* constructor */
method SendHandler
end-method;

method OnRequestSend
   /+ &MSG as Message +/
   /+ Returns Message +/
   /+ Extends/implements PS_PT:Integration:ISend.OnRequestSend +/
   /* Variable Declaration */
Local Boolean &bRet;


&bRet= &MSG.IBInfo.LoadConnectorProp("FILEOUTPUT");
&MSG.IBInfo.ConnectorOverride = True;
&bRet= &MSG.IBInfo.IBConnectorInfo.AddConnectorProperties
      ("sendUncompressed", "Y", %Header);
&bRet= &MSG.IBInfo.IBConnectorInfo.AddConnectorProperties
      ("FilePath", "c:\temp", %Property);

Return (&Msg);

End-Method;
```

 Copyright © 1988-2007, Oracle. All rights reserved.

The following example demonstrates overriding target connector properties using an implementation of the OnRequestSend method for a nonrowset-based asynchronous message.

```
import PS_PT:Integration:ISend;

class SendHandler implements PS_PT:Integration:ISend
   method SendHandler();
   method OnRequestSend(&Msg As Message) Returns Message;
end-class;

/* constructor */
method SendHandler
end-method;

method OnRequestSend
   /+ &MSG as Message +/
   /+ Returns Message +/
   /+ Extends/implements PS_PT:Integration:ISend.OnRequestSend +/
   /* Variable Declaration */

Local XmlDoc &xmldoc;
Local Boolean &bRet;


// if you have to access the content data for content based
// decisions, do this
&xmldoc = &MSG.GetXmlDoc();

&bRet= &MSG.IBInfo.LoadConnectorProp("FILEOUTPUT");
&MSG.IBInfo.ConnectorOverride = True;
&bRet= &MSG.IBInfo.IBConnectorInfo.AddConnectorProperties
      ("sendUncompressed", "Y", %Header);
&bRet= &MSG.IBInfo.IBConnectorInfo.AddConnectorProperties
      ("FilePath", "c:\temp", %Property);

Return (&MSG);

End-Method;
```

See Also

Enterprise PeopleTools 8.49 PeopleBook: PeopleCode API Reference, “Message Classes,” IBInfo Class

Enterprise PeopleTools 8.49 PeopleBook: PeopleCode API Reference, “Message Classes,” ConnectorInfo Collection

Enterprise PeopleTools 8.49 PeopleBook: PeopleCode API Reference, “Message Classes”

Copyright © 1988-2007, Oracle. All rights reserved.

Receiving and Processing Messages

This section discusses how to handle:

- Inbound asynchronous transactions.
- Inbound synchronous transactions.
- Simulating receiving messages from external nodes.

Note. The OnRouteReceive method can be implemented to apply PeopleCode that determines whether the default local node accepts the inbound message.

Handling Inbound Asynchronous Transactions

Implement the OnNotify method in the PS_PT application package, in the Integration sub-package, to handle inbound asynchronous transactions. All the examples in this section are assumed to be implementations of the OnNotify method. .

Message Class Inbound Asynchronous Example 1

The following example implements the OnNotify method.

```
import PS_PT:Integration:INotificationHandler;

class FLIGHTPROFILE implements PS_PT:Integration:INotificationHandler
   method FLIGHTPROFILE();
   method OnNotify(&_MSG As Message);
end-class;

/* constructor */
method FLIGHTPROFILE
end-method;

method OnNotify
   /+ &_MSG as Message +/
   /+ Extends/implements PS_PT:Integration:INotificationHandler.+/
   /+ OnNotify +/
   /* Variable Declaration */
   Local any &outstring;
   Local any &i;
   Local any &CRLF;

   Local Message &MSG;
   Local Rowset &rs, &rs1;
   Local Record &FLIGHTDATA, &REC;

   Local string &acnumber_value, &msi_sensor_value, &ofp_value,
   &actype_value, &callsign_value, &squadron_value, &comm1_value,
   &comm2_value, &ecm_value;

   Local XmlDoc &xmldoc;
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
Local string &return_string_value;
Local boolean &return_bool_value;

&CRLF = Char(13) | Char(10);

&MSG = &_MSG;

&return_bool_value = &MSG.IBInfo.ConnectorOverride;

For &i = 1 To &MSG.IBInfo.IBConnectorInfo.
GetNumberOfConnectorProperties()

   &return_string_value = &MSG.IBInfo.IBConnectorInfo.
   GetQueryStringArgName(&i);

   &return_string_value = &MSG.IBInfo.IBConnectorInfo.
   GetQueryStringArgValue(&i);

End-For;

&MSG.IBInfo.IBConnectorInfo.ClearConnectorProperties();

&return_string_value = &MSG.IBInfo.IBConnectorInfo.ConnectorName;
&return_string_value = &MSG.IBInfo.IBConnectorInfo.ConnectorClass
  Name;
&return_string_value = &MSG.IBInfo.IBConnectorInfo.Remote
  FrameworkURL;
&return_string_value = &MSG.IBInfo.IBConnectorInfo.PathInfo;
&return_string_value = &MSG.IBInfo.IBConnectorInfo.Cookies;

For &i = 1 To &MSG.IBInfo.IBConnectorInfo.GetNumberOfQuery
  StringArgs()

   &return_string_value = &MSG.IBInfo.IBConnectorInfo.
   GetConnectorPropertiesName(&i);

   &return_string_value = &MSG.IBInfo.IBConnectorInfo.
   GetConnectorPropertiesValue(&i);

   &return_string_value = &MSG.IBInfo.IBConnectorInfo.
   GetConnectorPropertiesType(&i);

End-For;

&MSG.IBInfo.IBConnectorInfo.ClearQueryStringArgs();

&return_string_value = &MSG.IBInfo.MessageType;
&return_string_value = &MSG.IBInfo.RequestingNodeName;
&return_string_value = &MSG.IBInfo.OrigUser;
&return_string_value = &MSG.IBInfo.OrigNode;
```

Copyright © 1988-2007, Oracle. All rights reserved.

```
   &return_string_value = &MSG.IBInfo.AppServerDomain;
   &return_string_value = &MSG.IBInfo.OrigProcess;
   &return_string_value = &MSG.IBInfo.OrigTimeStamp;
   &return_string_value = &MSG.IBInfo.DestinationNode;
   &return_string_value = &MSG.IBInfo.FinalDestinationNode;
   &return_string_value = &MSG.IBInfo.SourceNode;
   &return_string_value = &MSG.IBInfo.MessageName;
   &return_string_value = &MSG.IBInfo.MessageVersion;
   &return_string_value = &MSG.IBInfo.VisitedNodes;


   &rs = &MSG.GetRowset();
   &REC = &rs(1).QE_FLIGHTDATA;

   &FLIGHTDATA = CreateRecord(Record.QE_FLIGHTDATA);
   &REC.CopyFieldsTo(&FLIGHTDATA);

   /* Parse out Message Data */
   &acnumber_value = &FLIGHTDATA.QE_ACNUMBER.Value;
   &msi_sensor_value = &FLIGHTDATA.QE_MSI_SENSOR.Value;
   &ofp_value = &FLIGHTDATA.QE_OFP.Value;
   &actype_value = &FLIGHTDATA.QE_ACTYPE.Value;
   &callsign_value = &FLIGHTDATA.QE_CALLSIGN.Value;
   &squadron_value = &FLIGHTDATA.QE_SQUADRON.Value;
   &comm1_value = &FLIGHTDATA.QE_COMM1.Value;
   &comm2_value = &FLIGHTDATA.QE_COMM2.Value;
   &ecm_value = &FLIGHTDATA.QE_ECM.Value;

   &outstring = "Send Async FLight test";

   /* Construct Output String */
   &outstring = &outstring | &acnumber_value | &CRLF |
   &msi_sensor_value |
   &CRLF | &ofp_value | &CRLF | &actype_value | &CRLF |
   &callsign_value |
   &CRLF | &squadron_value | &CRLF | &comm1_value | &CRLF |
   &comm2_value |
   &CRLF | &ecm_value;

   /* Log Output String into page record */
   &FLIGHTDATA.GetField(Field.DESCRLONG).Value = &outstring;

   SQLExec("DELETE FROM PS_QE_FLIGHTDATA");
   &FLIGHTDATA.Insert();

end-method;
```

 Copyright © 1988-2007, Oracle. All rights reserved.

Message Class Inbound Asynchronous Example 2

The following example shows notification PeopleCode that checks the PSCAMA to determine the audit action code and that examines the language code to determine whether related language processing is needed:

```
method OnNotify
   /+ &MSG as Message +/

   /* Simple PeopleCode Notifcation- - Check the PSCAMA*/
   Local Rowset &MSG_RS;
   Local Record &REC_NAME_PREFIX, &REC, &REC_RL;
   Local integer &I;
   Local string &ACTION, &LNG, &BASE_LNG, &RELLNG, &KEY1, &KEY2;

   &MSG_RS = &MSG.GetRowset();
   For &I = 1 To &MSG_RS.RowCount /* Loop through all transactions */
      &REC = &MSG_RS.GetRow(&I).GetRecord(Record.NAME_PREFIX_TBL);
      /* Get Audit Action for this transaction */
      &ACTION = &MSG_RS.GetRow(&I).PSCAMA.AUDIT_ACTN.Value;
      /* Get Language code for this transaction */
      &LNG = &MSG_RS.GetRow(&I).PSCAMA.LANGUAGE_CD.Value;
      &BASE_LNG = %Language;

      Evaluate &ACTION /*****************************/
         /******** Add a Record *******/
         /*****************************/

      When "A"
         /* Add the base language record */
         &REC_NAME_PREFIX = CreateRecord(Record.NAME_PREFIX_TBL);
         &REC.CopyFieldsTo(&REC_NAME_PREFIX);
         &REC_NAME_PREFIX.ExecuteEdits();
         If &REC_NAME_PREFIX.IsEditError Then
            /* error handling */

         Else
            &REC_NAME_PREFIX.Insert();
            /* Need error handling here if insert fails */
            If &LNG <> %Language Then
               /* add related language record */
               &RELLNG = &REC.RelLangRecName;
               &REC_RL = CreateRecord(Record.NAME_PREFIX_LNG);
               &REC.CopyFieldsTo(&REC_RL);
               &REC_RL.LANGUAGE_CD.Value = &LNG;
               &REC_RL.Insert();

               /* Need error handling here if insert fails */
            End-If;
         End-If;

         /*****************************/
```

Copyright © 1988-2007, Oracle. All rights reserved.

```
   /***** Change a Record *******/
   /****************************/
   /**** Using record objects ***/

When "C"
   /* Get the Record - insert it */
   &KEY1 = &REC.GetField(Field.NAME_PREFIX).Value;
   &REC_NAME_PREFIX = CreateRecord(Record.NAME_PREFIX_TBL);

   &REC_NAME_PREFIX.NAME_PREFIX.Value = &REC.GetField(Field.
   NAME_PREFIX).Value;
   If &REC_NAME_PREFIX.SelectByKey() Then

      &REC.CopyFieldsTo(&REC_NAME_PREFIX);
      &REC_NAME_PREFIX.ExecuteEdits();
      If &REC_NAME_PREFIX.IsEditError Then
         /* error handling */
      Else
         &REC_NAME_PREFIX.Update();
      End-If;
   Else
      &REC.CopyFieldsTo(&REC_NAME_PREFIX);
      &REC_NAME_PREFIX.ExecuteEdits();
      If &REC_NAME_PREFIX.IsEditError Then
         /* error handling */
      Else
         &REC_NAME_PREFIX.Insert();
      End-If;
   End-If;

   /****************************/
   /****** Delete a Record ******/
   /****************************/
   /*** Using SQLExec ***********/

When "D"
   /* Get the Record using SQLExec- error */
   &KEY1 = &REC.GetField(Field.NAME_PREFIX).Value;
   SQLExec("Select NAME_PREFIX from PS_NAME_PREFIX_TBL where
   NAME_PREFIX = :⇒1", &KEY1, &KEY2);
   If None(&KEY2) Then
      /* Send to error log */
   Else
      SQLExec("Delete from PS_NAME_PREFIX_TBL where
      NAME_PREFIX = :1", &KEY1);

      SQLExec("Delete from PS_NAME_PREFIX_LNG where
      NAME_PREFIX = :1", &KEY1);
   End-If;
End-Evaluate;
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
   End-For;

end-method;
```

Message Class Inbound Asynchronous Example 3

The following example shows OnNotify PeopleCode with multiple transactions:

```
method OnNotify
   /+ &MSG as Message +/
   Local Rowset &HDR_RS, &LN_RS;
   Local Record &HDR_REC, &hdr_rec_msg, &LN_REC, &LN_REC_MSG;
   Local integer &I, &J;

   /*This notification peoplecode processes messages that may*/
   /*contain multiple Header (MSR_HDR_INV) transactions that may*/
   /*have multiple line (DEMAND_INF_INV) transactions. The data */
   /*is inserted into identically structured header/line tables*/
   /*(MSR_HDR_INV2 and DEMAND_INF_INV2)*/

   /* Create record objects for the Header and Lines that will be */
   /* inserted into */

   &HDR_REC = CreateRecord(Record.MSR_HDR_INV2);
   &LN_REC = CreateRecord(Record.DEMAND_INF_INV2);

   /* Create an App. Message Rowset that includes the
  MSR_HDR_INV (Header) and DEMAND_INF_INV (Line)*/
   &HDR_RS = &MSG.GetRowset();

   /* Clear out all the Headers and Lines */
   SQLExec("DELETE FROM PS_MSR_HDR_INV2 WHERE BUSINESS_UNIT =
   'M04A1'");
   SQLExec("DELETE FROM PS_DEMAND_INF_INV2 WHERE BUSINESS_UNIT =
   'M04A1'");

   /* Loop through all the headers in the message */
   For &I = 1 To &HDR_RS.ActiveRowCount
      /* Instantiate the row within the Header portion of the
/* App Message rowset to which data will be copied */
      &hdr_rec_msg = &HDR_RS.GetRow(&I).GetRecord(Record.MSR_HDR_INV);

      /* Copy data from the level 0 (Header portion) of
/* &STOCK_MSG message structure into the Header record object */
      &hdr_rec_msg.CopyFieldsTo(&HDR_REC);

      &HDR_REC.Insert();

      /* Create Rowset that includes the DEMAND_INF_INV (Line)
portion of the App Message Rowset */
```

Copyright © 1988-2007, Oracle. All rights reserved.

```
      &LN_RS = &HDR_RS(&I).GetRowset(1);
      /* Loop through all the lines within this header transaction */

      For &J = 1 To &LN_RS.ActiveRowCount
         /* Instantiate the row within the Line portion of the
/* App Message rowset to which data will be copied */
         &LN_REC_MSG = &LN_RS.GetRow(&J).GetRecord(Record.
         DEMAND_INF_INV);
         /* copy data into the Level 1 (Line portion) of &STOCK_MSG*/
         /* object */
         &LN_REC_MSG.CopyFieldsTo(&LN_REC);
         &LN_REC.Insert();
      End-For;
   End-For;
end-method;
```

Message Class Inbound Asynchronous Example 4

There's a practical limit to how large a message can be, and this can be controlled by a system-wide variable called *%MaxMessageSize*. The system administrator can change this size in the PSOPTIONS settings. This size can be set only for all messages, not for individual definitions.

PeopleCode that populates a Message object should include code that is similar to the following example to check the message size before inserting a new Level 0 row.

Note. Always code the %MaxMessageSize checkpoint at the Level 0 record. A batch of transactions can be split across multiple messages, but *do not* split a single transaction (logical unit of work) into multiple messages.

```
Local SQL &hdr_sql, &ln_sql;
Local Message &MSG;
Local Rowset &hdr_rs, &ln_rs;
Local Record &hdr_rec, &ln_rec, &hdr_rec_msg, &ln_rec_msg;

/* This PeopleCode will publish messages for a simple Header/
Line record combination. Multiple Header/Lines are copied to the
message until the %MaxMessageSize is exceeded at which point a
new messagis built. This references MSR_HDR_INV (Header) and
DEMAND_INF_INV (Line) records */

/* Create an instance of the STOCK_REQUEST message */
&MSG = CreateMessage(OPERATION.STOCK_REQUEST);

/* Create an App. Message Rowset that includes the
MSR_HDR_INV (Header) and DEMAND_INF_INV (Line)*/
&hdr_rs = &MSG.GetRowset();

/* Create a SQL object to select the Header rows */
&hdr_sql = CreateSQL("Select * from PS_MSR_HDR_INV
       WHERE BUSINESS_UNIT='M04A1'
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
 AND ORDER_NO LIKE 'Z%' ORDER BY BUSINESS_UNIT,ORDER_NO");
&I = 1;

/* Create record objects for the Header and Lines */
&ln_rec = CreateRecord(Record.DEMAND_INF_INV);
&hdr_rec = CreateRecord(Record.MSR_HDR_INV);

/* Loop through each Header row that is fetched */
While &hdr_sql.Fetch(&hdr_rec)
  /* Publish the message if its size exceeds the MaxMessageSize
  /* specified in Utilities/Use/PeopleTools Options */
  If &MSG.Size > %MaxMessageSize Then
   %IntBroker.Publish();
   &I = 1;
   /* Create a new instance of the message object */
   &MSG = CreateMessage(OPERATION.STOCK_REQUEST);
   &hdr_rs = &MSG.GetRowset();
  End-If;
  If &I > 1 Then
   &hdr_rs.InsertRow(&I - 1);
  End-If;
  /* Instantiate the row within the Header portion of the
   App Message rowset to which data will be copied */
  &hdr_rec_msg = &hdr_rs.GetRow(&I).GetRecord(Record.MSR_HDR_INV);
  /* Copy data into the level 0 (Header portion) of
  /* &MSG message structure */
  &hdr_rec.CopyFieldsTo(&hdr_rec_msg);

  /* Publish the last message if it has been changed*/
  If &hdr_rec_msg.IsChanged Then
   %IntBroker.Publish();
  End-If;
End-While;
```

Message Class Inbound Asynchronous Example 5

The following code example shows how to get data out of the IBInfo object for a rowset-based message.

```
Local Rowset &rs, &rs1;
Local Record &FLIGHTDATA, &REC;

Local string &acnumber_value, &msi_sensor_value, &ofp_value,
      &actype_value, &callsign_value, &squadron_value, &comm1_value,
      &comm2_value, &ecm_value, &datetime;
Local XmlDoc &xmldoc;
Local string &data;
Local boolean &return_bool_value;

&CRLF = Char(13) | Char(10);

/* this is how one would access data from IBinfo and
```

Copyright © 1988-2007, Oracle. All rights reserved.

```
/* IBConnectorInfo objects*/
&return_bool_value = &MSG.IBInfo.ConnectorOverride;

For &i = 1 To &MSG.IBInfo.IBConnectorInfo.GetNumberOfConnector
Properties()

   &data = &MSG.IBInfo.IBConnectorInfo.GetQueryStringArgName(&i);
   &data = &MSG.IBInfo.IBConnectorInfo.GetQueryStringArgValue(&i);

End-For;

&MSG.IBInfo.IBConnectorInfo.ClearConnectorProperties();

&data = &MSG.IBInfo.IBConnectorInfo.ConnectorName;
&data = &MSG.IBInfo.IBConnectorInfo.ConnectorClassName;
&data = &MSG.IBInfo.IBConnectorInfo.RemoteFrameworkURL;
&data = &MSG.IBInfo.IBConnectorInfo.PathInfo;
&data = &MSG.IBInfo.IBConnectorInfo.Cookies;

For &i = 1 To &MSG.IBInfo.IBConnectorInfo.GetNumberOfQueryStringArgs()

   &data = &MSG.IBInfo.IBConnectorInfo.GetConnectorPropertiesName(&i);
   &data = &MSG.IBInfo.IBConnectorInfo.GetConnectorPropertiesValue
     (&i);
   &data = &MSG.IBInfo.IBConnectorInfo.GetConnectorPropertiesType(&i);

End-For;

&MSG.IBInfo.IBConnectorInfo.ClearQueryStringArgs();

&data = &MSG.IBInfo.MessageType;
&data = &MSG.IBInfo.RequestingNodeName;
&data = &MSG.IBInfo.OrigUser;
&data = &MSG.IBInfo.OrigNode;
&data = &MSG.IBInfo.AppServerDomain;
&data = &MSG.IBInfo.OrigProcess;
&data = &MSG.IBInfo.OrigTimeStamp;
&data = &MSG.IBInfo.DestinationNode;
&data = &MSG.IBInfo.FinalDestinationNode;
&data = &MSG.IBInfo.SourceNode;
&data = &MSG.IBInfo.MessageName;
&data = &MSG.IBInfo.MessageVersion;
&data = &MSG.IBInfo.VisitedNodes;

/* get the content data from the message rowset*/
&rs = &MSG.GetRowset();
&REC = &rs(1).QE_FLIGHTDATA;

&FLIGHTDATA = CreateRecord(Record.QE_FLIGHTDATA);
&REC.CopyFieldsTo(&FLIGHTDATA);
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
/* Parse out Message Data */
&acnumber_value = &FLIGHTDATA.QE_ACNUMBER.Value;
&msi_sensor_value = &FLIGHTDATA.QE_MSI_SENSOR.Value;
&ofp_value = &FLIGHTDATA.QE_OFP.Value;
&actype_value = &FLIGHTDATA.QE_ACTYPE.Value;
&callsign_value = &FLIGHTDATA.QE_CALLSIGN.Value;
&squadron_value = &FLIGHTDATA.QE_SQUADRON.Value;
&comm1_value = &FLIGHTDATA.QE_COMM1.Value;
&comm2_value = &FLIGHTDATA.QE_COMM2.Value;
&ecm_value = &FLIGHTDATA.QE_ECM.Value;
&datetime = &FLIGHTDATA.ACTIONDTTM.Value;

&outstring = "Send Async FLight test";

/* Construct Output String */
&outstring = &outstring | &acnumber_value | &CRLF | &msi_sensor_value
     | &CRLF | &ofp_value | &CRLF | &actype_value | &CRLF
     | &callsign_value | &CRLF | &squadron_value | &CRLF
     | &comm1_value | &CRLF | &comm2_value | &CRLF | &ecm_value |
        &datetime;

/* Log Output String into page record */
&FLIGHTDATA.GetField(Field.DESCRLONG).Value = &outstring;

SQLExec("DELETE FROM PS_QE_FLIGHTDATA");
&FLIGHTDATA.Insert();
```

Message Class Inbound Asynchronous Example 6

The following code example shows how to get data out of the IBInfo object for a nonrowset-based message.

```
Local XmlDoc &xmldoc;
Local XmlNode &node, &root, &acct_id_node, &acct_name_node,
     &address_node, &phone_node;
Local string &outstring, &CRLF;
Local Record &FLIGHT_DATA_INFO, &REC;

Local string &data;
Local boolean &return_bool_value;

/* this is how one wouild access data from IBinfo and
/* IBConnectorInfo objects*/

&return_bool_value = &MSG.IBInfo.ConnectorOverride;

For &i = 1 To &MSG.IBInfo.IBConnectorInfo.GetNumberOfConnector
  Properties()

   &data = &MSG.IBInfo.IBConnectorInfo.GetQueryStringArgName(&i);
```

Copyright © 1988-2007, Oracle. All rights reserved.

```
   &data = &MSG.IBInfo.IBConnectorInfo.GetQueryStringArgValue(&i);

End-For;

&MSG.IBInfo.IBConnectorInfo.ClearConnectorProperties();

&data = &MSG.IBInfo.IBConnectorInfo.ConnectorName;
&data = &MSG.IBInfo.IBConnectorInfo.ConnectorClassName;
&data = &MSG.IBInfo.IBConnectorInfo.RemoteFrameworkURL;
&data = &MSG.IBInfo.IBConnectorInfo.PathInfo;
&data = &MSG.IBInfo.IBConnectorInfo.Cookies;

For &i = 1 To &MSG.IBInfo.IBConnectorInfo.GetNumberOfQueryStringArgs()

   &data = &MSG.IBInfo.IBConnectorInfo.GetConnectorPropertiesName(&i);
   &data = &MSG.IBInfo.IBConnectorInfo.GetConnectorPropertiesValue
     (&i);
   &data = &MSG.IBInfo.IBConnectorInfo.GetConnectorPropertiesType(&i);

End-For;

&MSG.IBInfo.IBConnectorInfo.ClearQueryStringArgs();

&data = &MSG.IBInfo.MessageType;
&data = &MSG.IBInfo.RequestingNodeName;
&data = &MSG.IBInfo.OrigUser;
&data = &MSG.IBInfo.OrigNode;
&data = &MSG.IBInfo.AppServerDomain;
&data = &MSG.IBInfo.OrigProcess;
&data = &MSG.IBInfo.OrigTimeStamp;
&data = &MSG.IBInfo.DestinationNode;
&data = &MSG.IBInfo.FinalDestinationNode;
&data = &MSG.IBInfo.SourceNode;
&data = &MSG.IBInfo.MessageName;
&data = &MSG.IBInfo.MessageVersion;
&data = &MSG.IBInfo.VisitedNodes;

&xmldoc = &MSG.GetXmlDoc();

&CRLF = Char(13) | Char(10);


&root = &xmldoc.DocumentElement;

/* Get values out of XMLDoc */
&node_array = &root.GetElementsByTagName("actype");
&ac_type_node = &node_array.Get(1);
&ac_type_value = &ac_type_node.NodeValue;

&node_array = &root.GetElementsByTagName("msi_sensor");
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
&msi_sensor_node = &node_array.Get(1);
&msi_sensor_value = &msi_sensor_node.NodeValue;

&node_array = &root.GetElementsByTagName("callsign");
&callsign_node = &node_array.Get(1);
&callsign_value = &callsign_node.NodeValue;

&node_array = &root.GetElementsByTagName("ofp");
&ofp_node = &node_array.Get(1);
&ofp_value = &ofp_node.NodeValue;

&outstring = "GetDataout of xmldoc Test";

&outstring = &outstring | &CRLF | &ac_type_value | &CRLF |
&msi_sensor_node
     | &CRLF | &callsign_value | &CRLF | &ofp_value;

/* Write out the result string */
SQLExec("DELETE FROM PS_QE_FLIGHT_DATA");
&FLIGHT_DATA_INFO = CreateRecord(Record.QE_FLIGHT_DATA);
&FLIGHT_DATA_INFO.GetField(Field.DESCRLONG).Value = &outstring;
&FLIGHT_DATA_INFO.Insert();
```

Message Class Inbound Asynchronous Example 7

The following example show a notification that could be an implementation of the OnNotify method, using a component interface in the notification. This example shows error trapping and has multilanguage support:

```
Component string &PUBNODENAME;
/* save pubnodename to prevent circular publishes */
Local Message &MSG;
Local Rowset &MSG_ROWSET, &cur_rowset;
Local ApiObject &oSession;
Local ApiObject &CONTACT_CI;
Local number &I;
Local string &KEY1;
Local Record &REC;
Local boolean &BC_CREATE, &ADD;
Local boolean &TRANSACTION_ERROR, &MSG_ERROR;
/** Transaction/Message Error Flags**/

Function errorHandler()
   Local ApiObject &oPSMessageColl;
   Local ApiObject &oPSMessage;
   Local string &strErrMsgSetNum, &strErrMsgNum, &strErrMsgText,
    &strErrType;
   &oPSMessageColl = &oSession.PSMessages;
   For &I = 1 To &oPSMessageColl.Count
      &oPSMessage = &oPSMessageColl.Item(&I);
      &strErrMsgSetNum = &oPSMessage.MessageSetNumber;
```

Copyright © 1988-2007, Oracle. All rights reserved.

```
      &strErrMsgNum = &oPSMessage.MessageNumber;
      &strErrMsgText = &oPSMessage.Text;
      &LogFile.WriteLine(&strErrType | " (" | &strErrMsgSetNum | ","
          | &strErrMsgNum | ") - " | &strErrMsgText);
   End-For;
   rem ***** Delete the Messages from the collection *****;
   &oPSMessageColl.DeleteAll();
End-Function;

Function DO_CI_SUBSCRIBE()

   &oSession = %Session;

   &CONTACT_CI = &oSession.GETCOMPONENT(CompIntfc.CONTACT);
   If (&CONTACT_CI = Null) Then
      /* Replace this message with Tools message set when available */
      Error MsgGet(91, 58, " Unable to get the Component Interface.");
      Exit (1);
   End-If;

   /** Set Component Interface Properties **/
   &CONTACT_CI.GetHistoryItems = True;
   &CONTACT_CI.Interactivemode = False; /** set this to True while
    debugging **/
   rem Send messages to the PSMessage Collection;
   &oSession.PSMessagesMode = 1;

   &MSG_ERROR = False;

   For &I = 1 To &MSG_ROWSET.ActiveRowCount

      /** Set Session Language Code Property **/
      &REGIONALSETTINGS = &oSession.RegionalSettings;
      &REGIONALSETTINGS.LanguageCd = &MSG_ROWSET(&I).PSCAMA.
        LANGUAGE_CD.Value;

      &TRANSACTION_ERROR = False;
      &BC_CREATE = False;

      /** Instantiate Component Interface **/
      &KEY1 = &MSG_ROWSET(&I).CONTACT_TBL.PERSON_ID.Value;
      &CONTACT_CI.PERSON_ID = &KEY1;

      Evaluate &MSG_ROWSET(&I).PSCAMA.AUDIT_ACTN.Value
      When = "A"
      When = "N"
         &ADD = True;

         /* Check if Keys already exist. */
         &CONTACT_CIColl = &CONTACT_CI.Find();
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
   /*If None(&EXISTS) Then*/
   If &CONTACT_CIColl.Count = 0 Then
      If &CONTACT_CI.Create() Then
         &BC_CREATE = True;
      Else
         /* Replace this message with Tools message set
            when available */
         Warning MsgGet(18022, 56, "Error creating Component
         Interface for transaction %1", &I);
         &TRANSACTION_ERROR = True;
      End-If;
   Else
      If Not &CONTACT_CI.Get() Then
         /* Replace this message with Tools message set
            when available */
         Warning MsgGet(18022, 59, "Could not Get Component
         Interface for transaction %1", &I);
         &TRANSACTION_ERROR = True;
      End-If;
   End-If;
   Break;
When = "C"
   &ADD = False;
   If Not &CONTACT_CI.Get() Then
      /* Replace this message with Tools message set when
         available */
      Warning MsgGet(18022, 59, "Could not Get Component
      Interface for transaction %1", &I);
      &TRANSACTION_ERROR = True;
   End-If;
   Break;
When = "D"
When = "K"
When-Other
   /* delete and old key action codes not allowed at this
      time */
   &TRANSACTION_ERROR = True;
   Warning MsgGet(18022, 61, "Audit Action 'D' not allowed on
   transaction %1", &TRANSACTION);
   Break;
End-Evaluate;

&CONTACT_CI.CopyRowset(&MSG_ROWSET, &I);

If Not &TRANSACTION_ERROR Then
   If Not &CONTACT_CI.save() Then
      /* Replace this message with Tools message set when
         available */
      Warning MsgGet(18022, 57, "Error saving Component
```

Copyright © 1988-2007, Oracle. All rights reserved.

```
            Interface for transaction %1", &TRANSACTION);
            &TRANSACTION_ERROR = True;
         End-If;
      End-If;

      /** close the last Component Interface in preparation for
          getting the next **/
      If Not &CONTACT_CI.Cancel() Then
         /* Replace this message with Tools message set when
            available */
         Warning MsgGet(18022, 58, "Error Canceling Component
         Interface for transaction %1", &TRANSACTION);
         Exit (1);
      End-If;

      /* Reset &TRANSACTION_ERROR to "False" for &BusComp.Save()
         to execute if no
      /*  Transaction Error found in the next Transaction.  */
      &TRANSACTION_ERROR = False;

   End-For;

   If &TRANSACTION_ERROR Then
      &MSG_ERROR = True;
   End-If;

End-Function;

/****  Main Process  ****/
&MSG.ExecuteEdits(%Edit_Required + %Edit_TranslateTable);
If &MSG.IsEditError Then
   &MSG_ERROR = True;
Else
   &PUBNODENAME = &MSG.PubNodeName;
   &MSG_ROWSET = &MSG.GetRowset();
   /* Do Component Interface subscribe */
   DO_CI_SUBSCRIBE();
End-If;

If &MSG_ERROR Then
   Exit (1);
End-If;
```

XmlDoc Class Inbound Asynchronous Example

The following example uses the GetXmlDoc method.

```
Local XmlDoc &BIGMAN;
Local XmlNode &node, &root;
Local string &outstring;
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
Local Rowset &LEVEL0;
Local Record &SALES_ORDER_INFO, &REC;

&CRLF = Char(13) | Char(10);

&BIGMAN = &MSG.GetXmlDoc();

&root = &BIGMAN.DocumentElement;
&child_count = &root.ChildNodeCount;

/* Get values out of XmlDoc */
&node_array = &root.GetElementsByTagName("QE_ACCT_ID");
&acct_id_node = &node_array.Get(2);
&account_id_value = &acct_id_node.NodeValue;

&node_array = &root.GetElementsByTagName("QE_ACCOUNT_NAME");
&acct_name_node = &node_array.Get(2);
&account_name_value = &acct_name_node.NodeValue;

&node_array = &root.GetElementsByTagName("QE_ADDRESS");
&address_node = &node_array.Get(2);
&address_value = &address_node.NodeValue;

&node_array = &root.GetElementsByTagName("QE_PHONE");
&phone_node = &node_array.Get(2);
&phone_value = &phone_node.NodeValue;

&outstring = "GetMessageXmlDoc Test";
&outstring = &outstring | &CRLF | &account_id_value | &CRLF
     | &account_name_value | &CRLF | &address_value | &CRLF |
       &phone_value;

&SALES_ORDER_INFO = CreateRecord(Record.QE_SALES_ORDER);
&SALES_ORDER_INFO.GetField(Field.QE_ACCT_ID).Value =
     &account_id_value;
&SALES_ORDER_INFO.GetField(Field.DESCRLONG).Value = &outstring;
&SALES_ORDER_INFO.Update();
```

Handling Inbound Synchronous Transactions

Implement the OnNotify method in the PS_PT application package, in the Integration sub-package, to handle inbound synchronous transactions. All the examples in this section are assumed to be implementations of the OnNotify method.

Message Class Inbound Synchronous Example

The following example implements both the OnNotify method and the OnError method.

```
import PS_PT:Integration:IRequestHandler;

class RequestMan implements PS_PT:Integration:IRequestHandler
```

Copyright © 1988-2007, Oracle. All rights reserved.

```
   method RequestMan();
   method OnRequest(&MSG As Message) Returns Message;
   method OnError(&MSG As Message) Returns string;
end-class;

/* constructor */
method RequestMan
   %Super = create PS_PT:Integration:IRequestHandler();
end-method;

method OnRequest
   /+ &MSG as Message +/
   /+ Returns Message +/
   Local Message &response;

   &response = CreateMessage(Operation.SYNC_REMOTE,
   %IntBroker_Response);

   &response.GetRowset().GetRow(1).GetRecord(Record.QE_FLIGHTDATA).
   GetField (Field.DESCRLONG).Value = &MSG.GenXMLString();

   Return &response;

end-method;

method OnError
   /+ &MSG as Message +/
   /+ Returns String +/
   /+ Extends/implements PS_PT:Integration:IRequestHandler.OnError +/
   Local integer &nMsgNumber, &nMsgSetNumber;
   Local string &sText;

   &nMsgNumber = &MSG.IBException.MessageNumber;
   &nMsgSetNumber = &MSG.IBException.MessageSetNumber;
   rem    &sText = &exception.DefaultText;
   &sText = &MSG.IBException.ToString();


   /* ADD SPECIFIC ERROR INFO HERE */
   Return &sText;

end-method;
```

XmlDoc Class Inbound Synchronous Example

The following example uses the GetXmlDoc method:

```
Local XmlDoc &xmlRequestDoc;
Local XmlDoc &xmlResponseDoc;
Local XmlNode &select;
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
Local Message &Return_MSG;
Local array of XmlNode &whereClause;

Local string &recordName;
Local string &fieldName;
Local string &fieldValue;
Local Rowset &outputRowset;
Local boolean &return_bool_value;

&xmlRequestDoc = &MSG.GetXmlDoc();
&select = &xmlRequestDoc.DocumentElement;

&recordName = &select.GetAttributeValue("record");
&outputRowset = CreateRowset(@("Record." | &recordName));

&whereClause = &select.GetElementsByTagName("where");
If &whereClause <> Null And
   &whereClause.Len <> 0 Then
  &fieldName = &whereClause.Get(1).GetAttributeValue("field");
  &fieldValue = &whereClause.Get(1).GetAttributeValue("value");
  &outputRowset.Fill("WHERE " | &fieldName | "= :1", &fieldValue);
Else
  &outputRowset.Fill();
End-If;

&Return_MSG = CreateMessage(OPERATION.EXAMPLE, %Int⇒
Broker_Response);
&xmlResponseDoc  = &Return_MSG.GetXmlDoc();
&b = &xmlResponseDoc.CopyRowset(&outputRowset);
Return &Return_MSG;
```

SoapDoc Class Inbound Synchronous Example

The following example uses the GetXmlDoc method.

Note. Because GetXmlDoc returns an XmlDoc object, you must receive the inbound request message as an XmlDoc object, then convert it to a SoapDoc object to process it with SOAP methods.

```
Local XmlDoc &request, &response;
Local string &strXml;
Local SoapDocs &soapReq, &soapRes;
Local Message &Response_Message;

&soapReq = CreateSoapDoc();

&request = &MSG.GetXmlDoc();
&soapReq.XmlDoc = &request;
&OK = &soapReq.ValidateSoapDoc();
&parmN = &soapReq.GetParmName(1);
&parmV = &soapReq.GetParmValue(1);
```

Copyright © 1988-2007, Oracle. All rights reserved.

```
&Response_Message = CreateMessage(OPERATION.SoapExample,
%IntBroker_Response);

&response  = &Response_Message.GetXmlDoc();
&soapRes = CreateSoapDoc();
&soapRes.AddEnvelope(0);
&soapRes.AddBody();
&soapRes.AddMethod("StockPrice", 1);
&soapRes.AddParm(&parmN, &parmV);
&soapRes.AddParm("Price", "29");
&OK = &soapRes.ValidateSoapDoc();

&response = &soapRes.XmlDoc;
Return &Response_Message;
```

Simulating Receiving Messages from External Nodes

You can use PeopleCode to simulate receiving asynchronous messages from external nodes, including running transformations.

Use can use the IntBroker class InboundPublish method to work with rowset-based and nonrowset-based messages.

The following example shows an inbound publish as part of an OnNotify method implementation with a rowset-based message:

```
Local Message &MSG_REMOTE;
Local Rowset &rs;

&rs = &MSG.GetRowset();
/*create the message to be re-published from a simualted remote node*/

&MSG_REMOTE = CreateMessage(OPERATION.QE_FLIGHTPLAN);

&MSG_REMOTE.CopyRowset(&rs);

%IntBroker.InBoundPublish(&MSG_⇒
REMOTE, Node.REMOTE_NODE);
```

The following example shows an inbound publish as part of an OnNotify implementation with a nonrowset-based message:

```
Local Message &MSG_REMOTE;
Local XmlDoc &xmldoc;
Local Rowset &rs;

&rs = &MSG.GetRowset();
/*create the message to be re-published from a simualted remote node*/

&MSG_REMOTE = CreateMessage(OPERATION.QE_FLIGHTPLAN);
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
/* populate the &xmldoc with data to be re-published*/

&MSG_REMOTE.SetXmlDoc(&xmldoc);

%IntBroker.InBoundPublish(&MSG_⇒
REMOTE, Node.REMOTE_NODE);
```

Processing Inbound Errors

This section discusses how to:

- Validate data.
- Use the Exit built-in function.
- Correct messaging errors.

Validating Data

You validate data differently depending on the PeopleCode class that you're using to receive the message.

XMLDoc Class Validation

You can validate incoming XML DOM-compliant messages by using the XmlDoc document type definition (DTD) that is delivered with your PeopleSoft application.

See *Enterprise PeopleTools 8.49 PeopleBook: PeopleCode API Reference*, "XmlDoc Class".

SoapDoc Class Validation

You can validate SOAP-compliant messages by using the ValidateSoapDoc method in the PeopleCode SoapDoc class.

See *Enterprise PeopleTools 8.49 PeopleBook: PeopleCode API Reference*, "SOAPDoc Class".

Message Class Validation

Have a message receiving process validate incoming data by:

- Using the ExecuteEdits method in the code to invoke the definitional edits.
- Calling PeopleCode validation built-in functions (if they already exist, for example in a FUNCLIB record, or if validation logic can be encapsulated within a small set of built-in functions) from within the receiving PeopleCode.
- Calling a component interface or Application Engine program from the receiving process (for complex validation logic).

  This enables you to reuse logic that is embedded in the component.

The ExecuteEdits method uses the definitional edits to validate the message. You can specify the following system variables alone or in combination. If you don't specify a variable, all of the edits are processed.

- %Edit_DateRange

Copyright © 1988-2007, Oracle. All rights reserved.

- %Edit_OneZero
- %Edit_PromptTable
- %Edit_Required
- %Edit_TranslateTable
- %Edit_YesNo

The following example processes all edits for all levels of data in the message structure:

```
&MYMSG.ExecuteEdits();
```

The following example processes the Required Field and Prompt Table edits:

```
&RECPURCHASEORDER.ExecuteEdits(%Edit_Required +
 %Edit_PromptTable);
```

ExecuteEdits uses *set processing* to validate data. Validation by using a component interface or a PeopleCode built-in function is usually done with row-by-row processing. If a message contains a large number of rows per rowset, consider writing the message to a staging table and calling an Application Engine program to do set processing if you want additional error checking.

ExecuteEdits sets several properties on several objects if there are any errors:

- IsEditError is set on the Message, Rowset, Row, and Record objects if any fields contain errors.
- EditError, MessageNumber, and MessageSetNumber are set on the Field object that contains the error.

If you don't want to use ExecuteEdits, you can set your own errors by using the field properties. Setting the EditError property to True automatically sets the IsEditError message property to True. You can also specify your own message number, message set number, and so on, for the field. If you use the Exit(1) built-in function, the message status changes to Error when you finish setting the fields that are in error.

Using the Exit Built-in Function

Use the Exit built-in function to invoke a messaging error process when the application finds an error. This works only when you use the PeopleCode Message class to process inbound transactions. The same error processing is invoked automatically if PeopleTools finds an unexpected error, such as a Structured Query Language (SQL) error. The Exit built-in function has an optional parameter that affects how the error is handled.

To handle error processing in application tables, use the Exit built-in function with no parameter, or just let the notification process finish normally. This marks the message receipt as successful and commits the data.

To handle the error tracking and correction with PeopleSoft Integration Broker, use the Exit built-in function with 1 as a parameter to log the errors, perform a rollback, and stop processing.

Using the Exit Built-in Function Without Parameters

In the `Exit( )` form (that is, Exit without any parameters specified), all data is committed and the message is marked as complete. Use this to indicate that everything processed correctly and to stop program processing. Note, though, that the message status is set to Complete; therefore, you can't detect or access errors in the Integration Broker Monitor. If errors did occur, the subscription code should write them to a staging table, and then you must handle all of the error processing.

The Exit built-in function:

- Sets the message status in the application message queue for the subscription to Done.

 Copyright © 1988-2007, Oracle. All rights reserved.

- Commits the transaction.
- Stops processing.

Following is an example of using Exit without a parameter:

```
&MSG.ExecuteEdits();
If &MSG.IsEditError then
  App_Specific_Error_Processing();
  Exit();
Else
  Specific_Message_Processing();
End-if;
```

Using the Exit Built-in Function with Parameters

When you supply a 1 as a parameter for the Exit built-in function, the function:

- Processes a rollback.
- Sets the message status in the message queue for the subscription to Error.
- Writes the errors to the subscription contract error table.
- Stops processing.

You can view all errors that have occurred for this message in the Integration Broker Monitor, even those errors that are detected by ExecuteEdits.

Following is an example of using the Exit function with 1 as a parameter:

```
&MSG.ExecuteEdits();
If &MSG.IsEditError then
  Exit(1);
Else
  Process_Message();
End-if;
```

See Also

Chapter 21, “Using the Service Operations Monitor,” page 413

Using Message Object Functionality With Nonrowset-Based Messages

Prior to the PeopleTools 8.44 release, there were two distinct paths for writing and executing PeopleCode for PeopleSoft Integration Broker which were based on whether you were working with rowset-based XML messages or nonrowset-based XML messages.

For rowset-based XML messages, you could use a rowset and all the properties and methods associated with the Message class. For nonrowset-based XML messages, you could not use the Message class, but instead used built-in functions such as PublishXmlDoc and GetMessageXmlDoc. In addition, when working with nonrowset-based messages and these built-in functions, you could only access content data.

Copyright © 1988-2007, Oracle. All rights reserved.

Effective with the PeopleTools 8.44 release, when working with nonrowset-based XML messages you can use *all* of the functionality of the Message object using two new methods, SetXMLDoc and GetXMLDoc.

SetXMLDoc	Use this method to load and pass nonrowset-based data into the Message object.
GetXMLDoc	Use this method to get nonrowset-based data out of the message object.

Using the SetXMLDoc Method

The following example shows how to use SetXMLDoc to use the Message object to publish a nonrowset-based message.

```
//&XmlDoc holds the nonrowset-based data as before.

// create an instance of the Message object
 &MSG = CreateMessage(OPERATION.QE_F18_ASYNC_XMLDOC);
// Load the Message object with the xmldoc data.
&MSG.SetXmlDoc(&XmlDoc);
// perform a publish for the nonrowset-based message
%IntBroker.Publish(&MSG);
```

Using the GetXMLDoc Method

The following code example shows how to use GetXMLDoc to get nonrowset-based XML out of the Message object.

```
Local XMLDOC &XmlDoc;

// get an xmldoc object loaded with the content data.
&XmlDoc = &MSG.GetXmlDoc();
```

See Also

Enterprise PeopleTools 8.49 PeopleBook: PeopleCode API Reference, "Message Classes"

Generating Test Messages

Use the Handler Tester utility to generate test messages.

See *Enterprise PeopleTools 8.49 PeopleBook: Integration Testing Utilities and Tools*

Working With Message Segments

This chapter provides an overview of message segments and discusses how to:

- Configure nodes to handle segmented messages.
- Create message segments.

 Copyright © 1988-2007, Oracle. All rights reserved.

- Delete message segments.
- Send and receive segmented messages.
- Access message segments.
- View message segment data.
- Use restartable processing for publishing large messages in batch.

Understanding Message Segments

When you create message segments, you can divide rowset-based and nonrowset-based messages into multiple data containers, or segments, for sending. Depending on the order in which you send a message that contains message segments, the receiving system can process the message as a whole, or process one segment at a time while the others are compressed in memory or held in the application database.

As a result creating message segments can enhance system performance and message exchange, especially when you are working with large messages that exceed one gigabyte (1 GB).

To create and manage message segments, you use several methods and properties of the PeopleCode Message class.

Understanding PeopleCode used to Work with Message Segments

This section discusses:

- Methods used with message segments.
- Properties used with message segments.

Methods Used with Message Segments

The following table lists the PeopleCode methods you can use when you work with message segments.

Method	Class	Description
CreateNextSegment	Message	Designates the end point of one segment and the beginning of a new segment.
DeleteOrphaned Segments	IntBroker	Used to delete segments that might have been orphaned if you were processing message segments using a PeopleSoft Application Engine program that had to be restarted.
DeleteSegment	Message	Deletes a segment.
GetSegment	Message	Gets the segment specified by the passed value. The passed value is the segment number.
UpdateSegment	Message	Use this method to update data within the current segment.

Note. Use the DeleteSegment and UpdateSegment methods only when storing segments data in memory. These methods do not function when segment data is stored in the database.

Copyright © 1988-2007, Oracle. All rights reserved.

Properties Used with Message Segments

The following table lists PeopleCode properties that you can use when you work with message segments.

Property	Class	Description
CurrentSegment	Message	Returns a number, indicating which segment is the current segment.
SegmentsUnOrder	IBInfo	Determines whether to process message segments in order or unordered. This property pertains to asynchronous messages only. The values are: • *True*: Process message segments unordered. • *False*: Process message segments in order. (Default.)
SegmentCount	Message	Returns the total number of segments in a message.
SegmentsByDatabase	Message	Enables you to override where message segment data is stored for a message. The values are: • *True*: Store message segments awaiting processing in the application database. • *False*: Store message segments awaiting processing in memory. (Default.)

See Also

Enterprise PeopleTools 8.49 PeopleBook: PeopleCode API Reference

Configuring Nodes to Handle Segmented Messages

This section describes how to configure nodes to handle segmented messages.

Understanding Configuring Nodes to Handle Segmented Messages

Before you can send segmented messages, you must configure the remote node defined on the local system to handle segmented messages by setting the Segment Aware option on the Node Definitions page in the PeopleSoft Pure Internet Architecture.

Warning! Do not set the Segment Aware option for remote PeopleSoft 8.45 or earlier nodes, or for third-party systems. If you do so, the receiving system will consume only the first segment of the messages and ignore any subsequent segments.

Configuring a Node to Handle Segmented Messages

To configure a node to handle segmented messages:

1. Select PeopleTools, Integration Broker, Integration Setup, Node Definitions.
2. Select a node with which to work and click OK.

   The Node Definitions page appears.

 Copyright © 1988-2007, Oracle. All rights reserved.

3. Select the Segment Aware box.
4. Click the Save button.

Creating Message Segments

This section provides an overview of creating message segments and message segment numbers and discusses how to:

- Create message segments.
- Count the number of segments in messages.
- Store message segments awaiting processing.
- Override where to store message segment awaiting processing.
- Specify the order in which to process message segments.
- Chunk asynchronous segmented messages.

Understanding Creating Message Segments

By default every message has one segment.

To create multiple message segments use the CreateNextSegment method in the location in the message where you want one segment to end and next segment to begin. Continue this process until you have created the desired number of segments for the message.

Segments can contain any number of rowsets of data (rowset-based messages) or rows of data (nonrowset-based messages).

Understanding Message Segment Numbers

When you create a message segment, PeopleSoft Integration Broker assigns a message segment number to the segment.

The first message segment has a message segment number or 1, and message segment numbers are increment by one sequentially thereafter. As an example, if you break a message into three segments, the first segment number is *1*, the second segment number is *2*, and the third segment number is *3*.

Creating Message Segments

The following example shows using the CreateNextSegment method to create three segments in the message QE_FLIGHTPLAN, populating each segment with data from the component buffer.

```
&MSG = CreateMessage(OPERATION.QE_FLIGHTPLAN);

&rs=&MSG.GetRowset();
//Now populate rowset
// End of first segment. Beginning of second segment.
&MSG.CreateNextSegment();

&rs=&MSG.GetRowset();
//Now populate rowset
//End of second segment. Beginning of third segment.
&MSG.CreateNextSegment();
```

Copyright © 1988-2007, Oracle. All rights reserved.

```
&rs=&MSG.GetRowset();
//Now populate rowset

%IntBroker.Publish(&MSG);
```

Counting the Number of Segments in Messages

You might have the need to determine the number of segments in a message. Use the SegmentCount property to determine this information.

Storing Message Segments Awaiting Processing

By default, message segments awaiting processing are stored in memory until all segments are processed. Once all segments are processed, PeopleSoft Integration Broker sends all data as one message.

Use the MessageSegmentFromDB parameter in PSADMIN to specify the number of segments to keep in memory before writing segmented messages to the database. The default value is *10*.

For synchronous messages, if the number of segments sent for processing exceeds the set for the MessageSegmentsFromDB parameter, an error occurs.

Overriding Where to Store Message Segments Awaiting Processing

You can override the number of segments to keep in memory before writing segmented messages to the database for a single message using the SegmentsByDatabase property of the Message class.

Storage Location	Description
Memory	When message segments are stored in memory, PeopleSoft Integration Broker writes all segments as one message to the database when you send the message. To store message segment data in memory, set the SegmentsByDatabase property to *False*. (Default.)
Application database	When message segments are stored in the database, PeopleSoft Integration Broker writes the segments to the database individually. When you store message segments in the database you can have an infinite number of segments in a message. To store message segment data in the application database, set the SegmentsByDatabase property to *True*.

When you store message segments in memory, the number of segments is limited by the value set in the MessageSegmentFromDB parameter in PSADMIN in the Setting for PUB/SUB servers section of the file.

When working with asynchronous messages, if you create more message segments then the value set, all segments are written to the database automatically and the SegmentsByDatabase property will automatically be set to True.

For synchronous messages, attempting to create more segments then the specified value will result in an error message.

Specifying the Order in Which to Process Message Segments

When you work with segmented asynchronous messages you can specify that PeopleSoft Integration Broker process the segments in order or unordered, using the SegmentsUnOrder property of the Message class.

 Copyright © 1988-2007, Oracle. All rights reserved.

Message Segment Processing	Description
In order	When Integration Broker processes message segments in order, it decompresses all message segments sequentially and then processes the message as a whole. In this situation, only one publication or subscription contract is created. To process message segment in order, set the SegmentsUnOrder property to *False*.
Unordered	When Integration Broker processes message segments unordered, it decompresses and processes all segments in parallel. In this situation, the system creates one publication or subscription contract for each message segment. To process message segment unordered, set the SegmentsUnOrder property to *True*.

If you attempt to send ordered segmented messages to a node that is not segment aware an error message will be created and can be viewed on the Message Errors tab on the Message Details page in Integration Broker Monitor.

See Chapter 21, "Using the Service Operations Monitor," page 413.

Chunking Asynchronous Segmented Messages

Chunking asynchronous segmented messages sends message in blocks to the receiving node.

When using chunking, messages instances display in *Hold* status in Integration Broker Monitor until all chunks are received. Once all chunks are received, the message status switches to *New*.

Note. Chunking applies to ordered asynchronous messages only.

The number of segments to chunk for an asynchronous message is determined by the value you set for the MessageSegmentByDatabase parameter in PSADMIN. The default value is *10*.

As an example, if a message has 20 segments and you set MessageSegmentByDatabase to *5*, PeopleSoft Integration Broker will send four groups (array of messages) of segments to the integration gateway, and each group will contain five segments.

Deleting Message Segments

You can delete message segments in a message only before you publish the message.

Use the DeleteSegment method of the Message class to perform the action.

You cannot delete the first segment in a message.

The following example demonstrates using the DeleteSegment method in an implementation of the OnRequestSend method.

```
import PS_PT:Integration:ISend;

class Send implements PS_PT:Integration:ISend
   method Send();
   method OnRequestSend(&message As Message) Returns Message;
```

Copyright © 1988-2007, Oracle. All rights reserved.

```
   method OnError(&message As Message)
end-class;

/* constructor */
method Send
   %Super = create PS_PT:Integration:ISend();
end-method;

method OnRequestSend
   /+ &message as Message +/
   /+ Returns Message +/
   /+ Extends/implements PS_PT:Integration:ISend.OnRequestSend +/
   Local integer &segment_number, &i;
   Local Rowset &rs;

   For &i = 1 To &message.SegmentCount
      &rs = Null;
      &message.GetSegment(&i);

      &rs = &message.GetRowset();

      /* determine that segment 3 needs to be deleted. */
      &segment_number = &i;

   End-For;

   &message.DeleteSegment(&segment_number);


   Return &message;
end-method;

method OnError
   /+ &message as Message +/
   /+ Extends/implements PS_PT:Integration:ISend.OnError +/

end-method;
```

Sending and Receiving Segmented Messages

To send a segmented message, use sending PeopleCode and events as you would with any other message.

To receive a segmented message, use notification PeopleCode or the implement the OnRequest method.

Accessing Segments in Messages

After you receive a segmented message, use the GetSegment method of the Message class to access message segment data.

 Copyright © 1988-2007, Oracle. All rights reserved.

After you access a message segment, use the Message class GetRowset or GetXmlDoc methods to work with the contents of the segment.

Warning! You can access only one segment in a message at a time. When you access a message segment, PeopleSoft Integration Broker removes the previously accessed message segment from memory.

When you access a message segment, set the existing rowset to null to eliminate storing multiple rowsets in the data cache.

The following example shows using the GetSegment method to access a message segment in the message QE_FLIGHTDATA.

```
For &i = 1 To &MSG.SegmentCount
   &rs = Null;  //Null the rowset to remove it from memory
   &MSG.GetSegment(&i);

   &rs = &MSG.GetRowset();
   &REC = &rs(1).QE_FLIGHTDATA;

   &FLIGHTDATA = CreateRecord(Record.QE_FLIGHTDATA);
   &REC.CopyFieldsTo(&FLIGHTDATA);

   /* Parse out Message Data */
   &acnumber_value = &FLIGHTDATA.QE_ACNUMBER.Value;
   &msi_sensor_value = &FLIGHTDATA.QE_MSI_SENSOR.Value;
   &ofp_value = &FLIGHTDATA.QE_OFP.Value;
   &actype_value = &FLIGHTDATA.QE_ACTYPE.Value;
   &callsign_value = &FLIGHTDATA.QE_CALLSIGN.Value;
   &squadron_value = &FLIGHTDATA.QE_SQUADRON.Value;
   &comm1_value = &FLIGHTDATA.QE_COMM1.Value;
   &comm2_value = &FLIGHTDATA.QE_COMM2.Value;
   &ecm_value = &FLIGHTDATA.QE_ECM.Value;

   &outstring = "Send Async Flight test";

   /* Construct Output String */
   &outstring = &outstring | &acnumber_value | &CRLF |
   &msi_sensor_value | &CRLF | &ofp_value | &CRLF | &actype_value |
   &CRLF | &callsign_value | &CRLF | &squadron_value | &CRLF |
   &comm1_value | &CRLF | &comm2_value | &CRLF | &ecm_value;

   /* Log Output String into page record */
   &FLIGHTDATA.GetField(Field.DESCRLONG).Value = &outstring;

   SQLExec("DELETE FROM PS_QE_FLIGHTDATA");
   &FLIGHTDATA.Insert();

End-For;
```

Copyright © 1988-2007, Oracle. All rights reserved.

Viewing Message Segment Data

The Integration Broker Monitor Message Details page provides information about messages that contain segments.

See Also

Chapter 21, “Using the Service Operations Monitor,” page 413

Using Restartable Processing for Publishing Large Messages in Batch

This section provides an overview, prerequisites and setup steps for using restartable processing for publishing large asynchronous segmented messages in batch.

Understanding Using Restartable Processing

PeopleSoft provides a PeopleSoft Application Engine library module, IB_SEGTEST, that you can use as a template to create a module to aid in processing large messages and messages in batch for outbound asynchronous PeopleSoft Integration Broker segment data with restart capability.

With restart capability, if there is an abnormal program termination, you can correct any data errors and continue processing from the point of the last commit without having to reload message segment data from the beginning.

Understanding the IB_SEGTEST Application Engine Library Module

This section provides overview information for using the IB_SEGTEST

The IB_SEGTEST library module consists of three sections:

- Section 1: Section1. The main processing section.
- Section 2: ABORT. Use to trigger a user abort of the running application engine program
- Section 3: CLEANSEG. An independent section you can call to clean up pending segment data that had been committed to the database but is no longer to be used.

Prerequisites

To use the information provided in this section, you should have a thorough understanding of PeopleSoft Application Engine.

Using the IB_SEGTEST Library Module

This section provides an overview of the high-level list of tasks to perform to set up a PeopleSoft Application Engine program to perform restartable message processing.

1. Make a copy of IB_SEGTEST, including all sections and PeopleCode.

   From here on, the copy of the application engine library module is referred to as IB_SEGTEST1, but you can use any name you choose.

2. In the State Records tab of IB_SEGTEST1, verify that PSIBSEGRSTR_AET is the default state record. Replace PT_EIP_ERR_AET with whatever state record is used in the main application engine program that will be calling the Library module.

 Copyright © 1988-2007, Oracle. All rights reserved.

Note that IB_SEGTEST1 is flagged as not restartable. Since database commits will be performed in the middle of PeopleCode processing, the only way the commits can take effect is if the module is flagged as not restartable.

3. The application engine program used to call IB_SEGTEST1 should be restartable.

   Always issue a commit in the step prior to calling the library module IB_SEGTEST1.

4. In the application engine program that will be calling IB_SEGTEST1, insert a step to call IB_SEGTEST1, section Section1. Insert the step at the point in time when you want to do the message publish. You must issue a commit prior to calling this section, otherwise there will be a 'Unable to Process Commit' error issued from within IB_SEGTEST1.
5. Add PSIBSEGRSTR_AET as an additional state record to the calling application engine program.
6. Since both programs now share state records, when IB_SEGTEST1 is called, all state record values will be passed on to the called module. Presumably all application values needed to extract application data would be stored in the application state record.
7. Modify the PeopleCode in IB_SEGTEST1.Section1. Several comments have been added to the code to aid in the modifications. Note the following:
   - Change `&MSG = CreateMessage(OPERATION.QE_FLIGHTPLAN)` to create whatever message will be used.
   - SegmentsByDatabase should always be set to True.
   - The While loop is used to simulate application code processing large volumes of data. This can be changed to meet application needs. However, pay close attention as to when commits are issued, when state records are updated, when new segments are created, and finally, when the message publish is executed. The order of these events is crucial to proper workability. In the sample program, also note how to break out of the While loop.
   - Note the location where the application state record needs to be updated. A comment instructs in the PeopleCode provides instructions on where to perform this task.
   - Do not remove the `Exit(1)` from the end of the PeopleCode. This is necessary to bypass the Abort action that is coded into the same Step.
   - If in the middle of processing, the application code determines that an abort needs to be triggered, an `Exit(0)` can be coded. This triggers the Abort step to be called, which will terminate application engine processing. A restart could then be issued if processing needs to continue.

     If you determine that a message no longer needs to be published, the calling application engine program could then call the CLEANSEG step to get rid of all the pending data that has been saved in the database. Alternatively, the Abort step could be modified to call CLEANSEG if on any abort, no old data is to be kept.

See Also

Enterprise PeopleTools 8.49 PeopleBook: PeopleSoft Application Engine

Copyright © 1988-2007, Oracle. All rights reserved.

Copyright © 1988-2007, Oracle. All rights reserved.

CHAPTER 13

Building Message Schemas

This chapter provides an overview of the message Schema Builder and describes how to:

- Select and view data in the message Schema Builder.
- Build message schemas for rowset-based messages.
- Import message schemas for nonrowset-based messages.
- Modify message schemas.
- Delete message schemas.

Understanding the Message Schema Builder

The message Schema Builder enables you to build, import, modify and delete XML message schemas.

Note. The terms *message schema*, *XML message schema*, and *schema* are used interchangeably in this chapter.

To test message schemas during development, use the Schema Tester utility.

To enable runtime schema validation, use the Service Operations - General page to enable runtime validation for a service operation, or use the Service Schema Validation page to enable validation for several service operations at a time.

See Also

Enterprise PeopleTools 8.49 PeopleBook: PeopleSoft Integration Testing Utilities and Tools, "Using the Schema Tester Utility"

Chapter 16, "Enabling Runtime Message Schema Validation," page 315

Message Schemas

An *XML message schema* describes a model for the arrangement of tags and text in a valid XML document. A schema provides a common vocabulary for a particular application that exchanges documents.

Building, Importing, Modifying and Deleting Message Schemas

You can use the Message Schema Builder to manage message schemas for rowset-based messages in the application database.

Copyright © 1988-2007, Oracle. All rights reserved.

Note. You can also use the pages of the Message Builder component to manage rowset-based and nonrowset-based schemas. However, the Message Builder enables you to work with only one message schema at a time, whereas , the Message Schema Builder enables you to perform actions, such as building and deleting message schemas, on multiple messages at a time.

Note. You cannot use the Message Schema Builder to build schemas for message parts or container messages. You must use the Message Builder component to build schemas for these message types.

Rowset-Based Message Schemas

Use the Message Schema Builder to generate, regenerate, view or delete rowset-based message schemas.

You cannot regenerate or delete a rowset-based message schema that is a message part. Part and container schemas are automatically generated at save time so there's no need to explicitly regenerate or delete them.

Nonrowset-Based Message Schemas

Use the Message Schema Builder to import new nonrowset-based schemas into the database, modify existing nonrowset-based message schemas, or delete them.

Schemas for nonrowset-based message parts can be deleted or modified, but message parts should never be without a schema. After deleting a nonrowset-based message part, you should always import or enter a new schema for the message.

Selecting and Viewing Data in the Message Schema Builder

This section discusses how to:

- Select data in the Message Schema Builder.
- View message schema data details.
- View XML message schema code.

Pages Used To Select and View Data in the Message Schema Builder

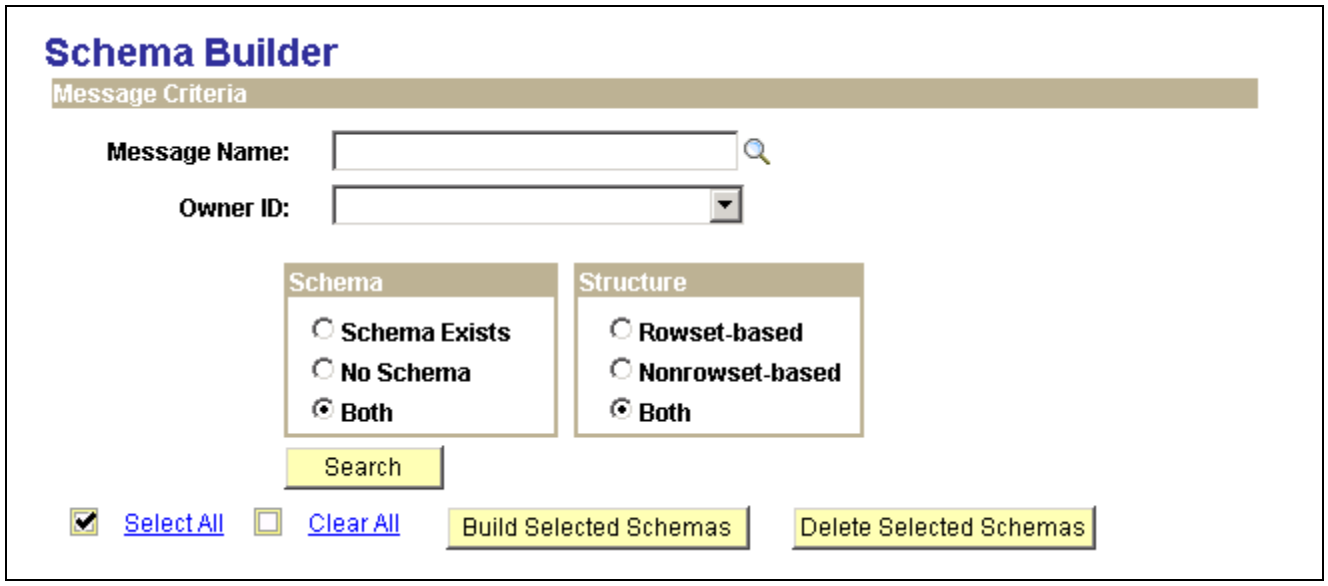
Page Name	Object Name	Navigation	Usage
Schema Builder page	IB_SCHEMABUILD	Select PeopleTools, Integration Broker, Service Utilities, Message Schema Builder.	Use this page to: • Search for messages and schemas with which to work. • View search results.
Schema Viewer page	IB_SCHEMABUILD_SEC	Click a message on the Schema Builder page.	View the XML schema in the schema viewer.

 Copyright © 1988-2007, Oracle. All rights reserved.

Selecting Data in the Message Schema Builder

When you access the Message Schema Builder component (IB_SCHEMABUILD) the Schema Builder page displays a search engine to use to search for messages and message schema data with which to work and view.

To access the Schema Builder page, select PeopleTools, Integration Broker, Service Utilities, Message Schema Builder.



Schema Builder page

The Schema Builder page provides the following options for searching for data with which to work and view in the application database.

Message Name (Optional.) Click the Lookup button to locate a message definition with which to work.

If you do not select a message name, the search will be based on all messages definitions in the application database.

Owner ID (Optional.) From the Owner ID dropdown list, select the owner ID for the message definition.

The owner ID helps to determine the application team that last made a change to a message definition. The values in the dropdown list box are translate table values that you can define in the OBJECTOWNERID field record.

Schema Select from the following options in the Schema group box:

- Schema Exists. Select this option to search message versions for which schemas have been built.
- No Schema. Select this option to search message versions for which no schemas have been built.
- Both. (Default.) Select this option to search all message versions.

Structure Select from the following options in the Structure group box:

- Rowset-based. Select this option to search for rowset-based message versions.

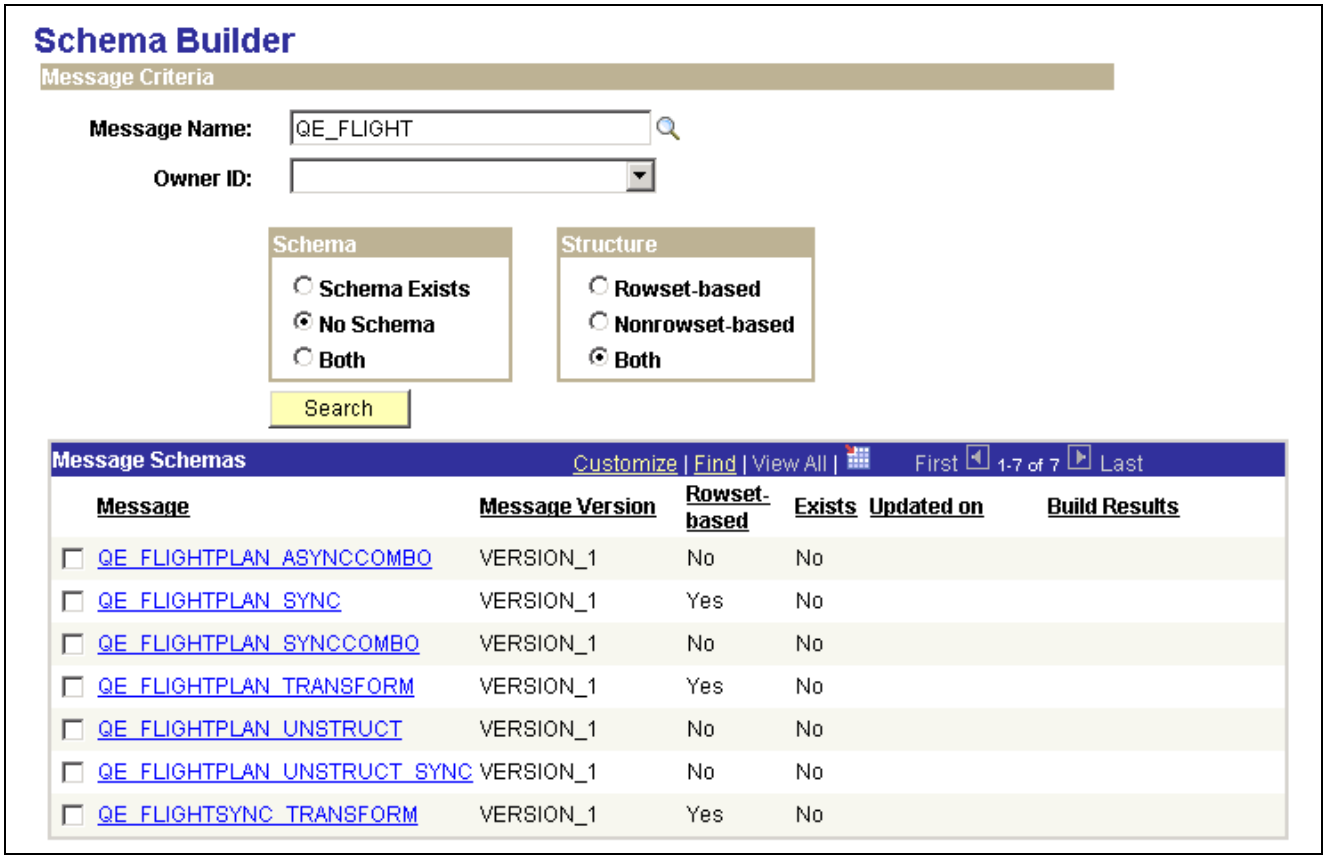
Copyright © 1988-2007, Oracle. All rights reserved.

- Nonrowset-based. Select this option to search for nonrowset-based message versions.
- Both. (Default.) Select this option to search for rowset-based and nonrowset-based message versions.

Search Click the button to search the database based on the criteria selected.

Viewing Message Schema Details

When you search for data in the Schema Builder, message detail results appear in the Message Schemas grid.



Message schemas grid

Message Message name returned from the search of the application database.

Message Version Version of the message returned from the search of the application database.

Rowset-based Indicates the structure of the message. The valid values are:

- *Yes*. Indicates that the message is a rowset-based message.
- *No*. Indicates that the message is a nonrowset-based message.

Exists Indicates whether a schema has been built for the message. The valid values are:

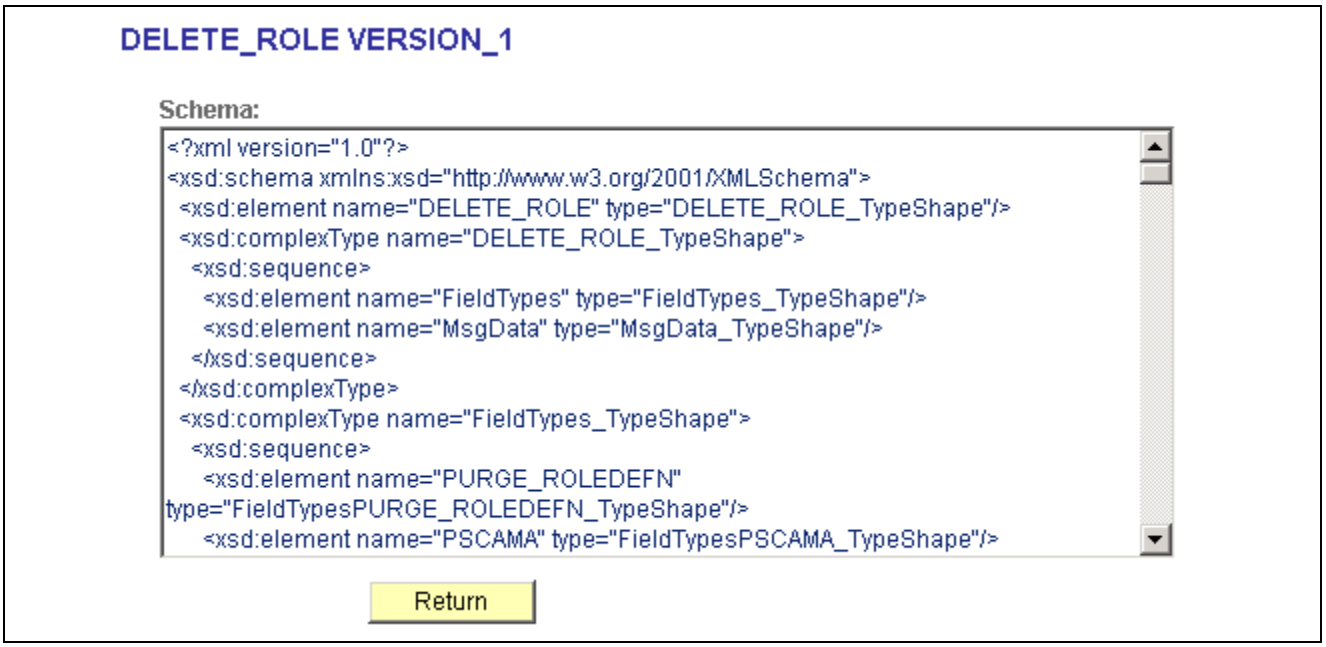
- *Yes*. A schema has been built for the message.
- *No*. A schema has not been built for the message.

Copyright © 1988-2007, Oracle. All rights reserved.

Updated On	Timestamp of the last update of the record. A new timestamp displays when a schema is generated or deleted for a message.
Build Results	Displays the results of actions performed on a schema.
Build Selected Schemas	Click the button to build schemas for the selected messages.
Delete Selected Schemas	Click the button to delete schemas that exist for the selected messages.

Viewing XML Message Schema

If a message schema exists for a message, click the message name in the Message Schema grid to view the schema details.



Schema details for version 1 of the DELETE_ROLE message definition.

Note. For easier viewing, highlight the data with your cursor.

Message schemas for rowset-based messages are read-only. You can edit message schemas for nonrowset-based messages.

Building Message Schemas for Rowset-Based Messages

This section discusses how to build message schemas for rowset-based messages.

Copyright © 1988-2007, Oracle. All rights reserved.

Page Used to Build Message Schemas for Rowset-Based Messages

Page Name	Object Name	Navigation	Usage
Schema Builder page	IB_SCHEMABUILD	Select PeopleTools, Integration Broker, Service Utilities, Message Schema Builder.	Use this page to build message schemas for rowset-based messages.

Building a Message Schema for a Rowset-Based Message

To build a message schema for a rowset-based message:

1. Access the Schema Builder page.
2. Search the application database for the message or messages for which to build schemas.

   See Chapter 13, "Building Message Schemas," Selecting Data in the Message Schema Builder, page 267.
3. Check the box next to the message or messages for which to build schemas.
4. Click the Build Selected Schemas button.

When the schema is built successfully, a timestamp appears in the Updated On field and the Build Results field displays *Successful Schema Insert*.

Importing Message Schemas for Nonrowset-Based Messages

This section discusses how to import message schemas for nonrowset-based messages.

Pages Used to Import Message Schemas for Nonrowset-Based Messages

Page Name	Object Name	Navigation	Usage
Schema Builder page	IB_SCHEMABUILD	Select PeopleTools, Integration Broker, Service Utilities, Message Schema Builder.	Use this page to locate and select the message for which to import a schema.
Schema page	IB_SCHEMABUILD_SEC	In the Message Schema results grid on the Schema Builder page, click the name of a nonrowset-based message.	Import or enter XML schema for a nonrowset-based message.

Importing a Message Schema for a Nonrowset-Based Message

To import schemas for nonrowset-based messages:

1. Access the Schema Builder page.

 Copyright © 1988-2007, Oracle. All rights reserved.

2. Use the Message Schema Builder search engine to locate the message for which you want to import a schema.

   See Chapter 13, "Building Message Schemas," Selecting Data in the Message Schema Builder, page 267.

3. In the Message Schema grid, click the message name link for the message for which you want to import a schema.
4. Import the schema.
   - Import a schema from a file.

     You can import a schema from a file by using the Upload Schema from File button and selecting the file to import. After you import the file, the contents displays in the Schema text box.

     Note. If you receive the error, "Error retrieving the file from database," verify that one of the variables PS_FILEDIR or PS_SERVDIR is defined in the system variables on your machine.

   - Direct data entry.

     You can also enter the schema directly in the Schema text box.
5. Click the Save button.

   The Schema Builder page appears.

A timestamp appears in the Updated On field and the Build Results field displays *Successful Schema Insert*.

Modifying Message Schemas

This section discusses how to modify message schemas.

Note. You can modify the content of message schemas built for nonrowset-based messages only.

To modify a schema, you can edit it directly in the Message Schema Builder, or you can export to make changes.

Pages Used to Modify Message Schemas

Page Name	Object Name	Navigation	Usage
Schema Builder page	IB_SCHEMABUILD	Select PeopleTools, Integration Broker, Service Utilities, Message Schema Builder.	Use this page to locate and select the message that contains the message schema to modify.
Schema page	IB_SCHEMABUILD_SEC	In the Message Schema results grid on the Schema Builder page, click the name of a nonrowset-based message.	Import a new message schema or modify the existing XML schema for a nonrowset-based message.

Modifying a Message Schema

To modify a message schema:

Copyright © 1988-2007, Oracle. All rights reserved.

1. Use the Message Schema Builder search engine to locate the message with which you want to work.

   See Chapter 13, "Building Message Schemas," Selecting Data in the Message Schema Builder, page 267.

2. In the Message Schema grid, click the message name link.

   A new page displays with the message schema populated in a text box.

3. Modify the schema as needed.

   - Modify the schema directly in the text box, or
   - Modify the schema in the editor of your choice.

     Use your cursor to highlight the contents of the text box and use the keyboard command CTRL + C to copy the contents of the text box. Paste the contents into your editor using the keyboard command CTRL + V. Modify the content as needed. Import the content back into the Message Schema Builder using the instructions described previously in this chapter for importing message schemas for nonrowset-based messages.

     See Chapter 13, "Building Message Schemas," Importing Message Schemas for Nonrowset-Based Messages, page 270.

4. Click the Save button.

The Schema Builder page displays and the Updated On field displays the date and time of the modification, and the Build Results field displays the results of the new schema build.

Deleting Message Schemas

This section discusses how to delete message schemas.

Understanding Deleting Message Schemas

You can delete message schemas using the Message Schema Builder page in the Message Schema Builder component (IB_SCHEMABUILD) or using the Message Schemas page in the Service Administration component (IB_HOME_PAGE).

Note. The Message Schema Builder page provides more comprehensive capabilities for searching for message schema.

You cannot delete a message schema when the message on which the schema is based is:

- Referenced in a service operation.
- Referenced as a message part in a container message.
- A rowset-based message part.
- A container message.
- Referenced in an provided WSDL document.

 Copyright © 1988-2007, Oracle. All rights reserved.

Page Used to Delete Message Schemas

Page Name	Object Name	Navigation	Usage
Schema Builder page	IB_SCHEMABUILD	Select PeopleTools, Integration Broker, Service Utilities, Message Schema Builder.	Delete message schemas.
Message Schemas	IB_HOME_PAGE6	Select PeopleTools, Integration Broker, Service Utilities, Service Administrations. Click the Message Schemas tab.	Delete message schemas.

Using the Message Schema Builder Page to Delete Message Schemas

When deleting a schema using the Message Schema Builder page use only the Delete Selected Schemas button. Do attempt to delete message schemas by deleting content in the Schema text box in the schema details view; if you save the changes, PeopleSoft Integration Broker will attempt to validate the blank schema at runtime.

To delete a message schema:

1. Access the Message Schema Builder page.
2. Use the Message Schema Builder search engine to locate the message with which you want to work.

   See Chapter 13, "Building Message Schemas," Selecting Data in the Message Schema Builder, page 267.

   The Schema Builder page appears.
3. In the Message Schema section, check the boxes next to the message names that contain schemas you want to delete.
4. Click the Delete Selected Schemas button.

Using the Message Schemas Page to Delete Message Schemas

The service system status that is set on the Service Configuration page affects the ability to delete message schemas.

See Chapter 14, "Managing Services," Understanding Configuring PeopleSoft Integration Broker for Handling Services, page 277.

To use the Message Schemas page to delete a message schema:

1. Access the Message Schemas page.
2. In the Message Name field, enter the name of the message that contains the schema to delete.
3. Click the Search button.

   The results of the search appear in the Messages with Schema section.
4. Check the box next to each message that contains a schema to delete.
5. Click the Delete button.

Copyright © 1988-2007, Oracle. All rights reserved.

 Copyright © 1988-2007, Oracle. All rights reserved.

CHAPTER 14

Managing Services

This chapter provides an overview of managing services and discusses how to:

- Configure PeopleSoft Integration Broker for handling services.
- Specify UDDI repositories in the PeopleSoft system.
- Access and view service definitions.
- Add service definitions.
- Configure service definitions.
- Restrict write access to services.
- Rename and delete services.

Understanding Managing Services

Services are used to logically group a set of service operations.

For example, if you have a number of service operations that are related to customers, such as those pertaining to customer information, adding customers, updating customers, deleting customers, and so on, you can create a customer web service and then associate the related service operations with that service.

Warning! PeopleSoft delivers two services with PeopleSoft Integration Broker: IB_GENERIC and IB_UTILITY. These services are used internally by the system. Do not delete or modify these services.

Common Elements Used in This Chapter

Comments (Optional.) Enter comments about the service or service definition.

Description Description of the service.

Generate SOAP Template Click to open the Generate SOAP Template utility. The utility enables you to generate SOAP documents for each service operation in a service for testing purposes.

Object Owner ID (Optional.) Indicates the owner of the service.

The owner ID helps to determine the application team that last made a change to a service definition. The values in the dropdown list box are translate table values that you can define in the OBJECTOWNERID field record.

Copyright © 1988-2007, Oracle. All rights reserved.

Operation Type Specifies how the service is transmitted.

On the Service page this field defines the operation type of the service operation added.

Provide Web Service Click to launch the Provide Web Services component and export PeopleSoft services as WSDL documents.

Schema Namespace Provides qualification for attributes and elements within an XML schema document (XSD).

The default is *http://xmlns.oracle.com/Enterprise/Tools/schemas.*

The namespace on the message definition defaults to the schema namespace you set as the default on the Service Configuration page.

Note. If you change the namespace, all future messages will have the new namespace.

Service The name of the service.

Service Alias (Optional.) Overrides the service name and will be the name of the service when the WSDL is provided or exported. The alias enables you to use mixed case in the name.

Service Operation The name of the service operation that is associated with the service.

On the Services page, use this field to add new service operations for the current service.

Service Namespaceand**Namespace** The service namespace field in the Service Configuration page used to default the service namespace field when defining PeopleSoft services. The default value is *http://xmln.oracle.com/enterprise/tools/service*.

The namespace field on the Service pages provides qualification for attributes and elements within a WSDL document.

Service System Status The status that is selected restricts rename, delete, and other administrative actions that users can perform on integration metadata in the Services Administration component.

Values are:

- *Production.*
- *Development.*

See Chapter 14, “Managing Services,” Understanding Configuring PeopleSoft Integration Broker for Handling Services, page 277.

Target Location Specifies the URL to be used for service requests.

You must define this location before creating services and service schemas.

View WSDL Click to view WSDL documents that were generated for the service in the WSDL repository.

 Copyright © 1988-2007, Oracle. All rights reserved.

Configuring PeopleSoft Integration Broker for Handling Services

Before you can work with services in PeopleSoft Integration Broker, you must set several configuration properties.

This section provides an overview of configuring PeopleSoft Integration Broker for handling services and discusses how to set service configuration properties.

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.

Service System Status

The Services Configuration page contains a Service System Status dropdown 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 dropdown 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

Copyright © 1988-2007, Oracle. All rights reserved.

Object	Action	Production Mode	Development Mode
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.
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.

Copyright © 1988-2007, Oracle. All rights reserved.

Object	Action	Production Mode	Development Mode
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.

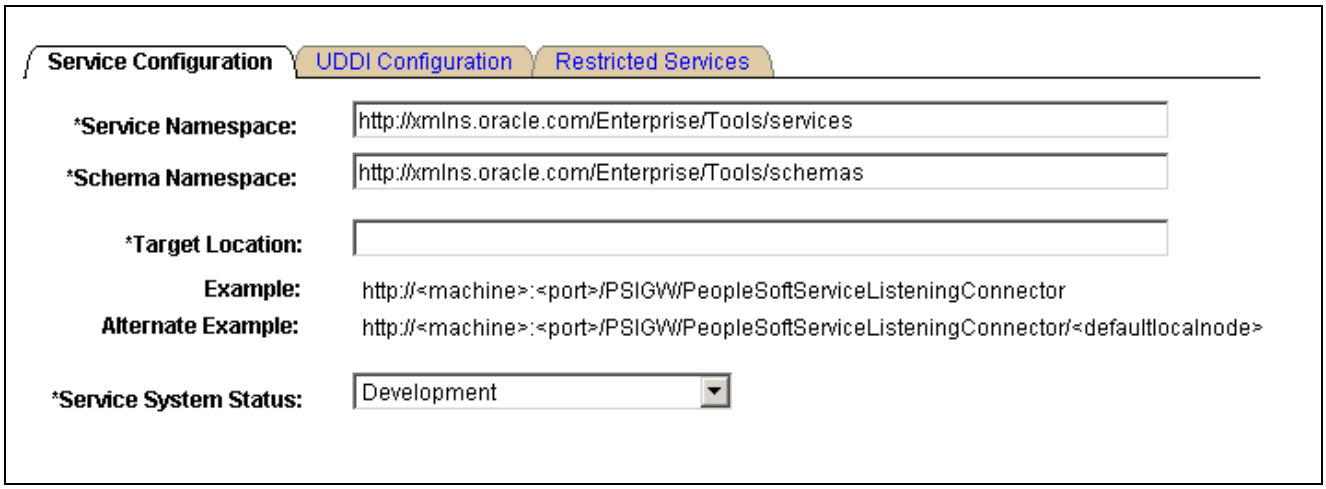
Page Used to Configuring PeopleSoft Integration Broker for Handling Services

Page Name	Object Name	Navigation	Usage
Service Configuration page	IB_SVCSETUP	Select PeopleTools, Integration Broker, Integration Setup, Service Configuration.	Use this page to configure PeopleSoft Integration Broker for handling services.

Setting Service Configuration Properties

You set service configuration properties on the Service Configuration page 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:

Copyright © 1988-2007, Oracle. All rights reserved.



Service Configuration page

To set service configuration properties:

1. Access the Service Configuration Properties page.
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 a URL to be used for service requests.

   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 target location URL format of http://<machine>/PSIGW/PeopleSoftServiceListeningConnector/<defaultnode>.
5. From the Service System Status dropdown list box, select one of the following options:
   - *Development.* (Default.)
   - *Production*.

   See Chapter 14, "Managing Services," Understanding Configuring PeopleSoft Integration Broker for Handling Services, page 277.
6. Click the Save button.

Specifying UDDI Repositories in the PeopleSoft System

This section discusses how to specifying UDDI repositories in the PeopleSoft system for providing and consuming services.

Understanding Specifying UDDI Repositories in the PeopleSoft System

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.

 Copyright © 1988-2007, Oracle. All rights reserved.

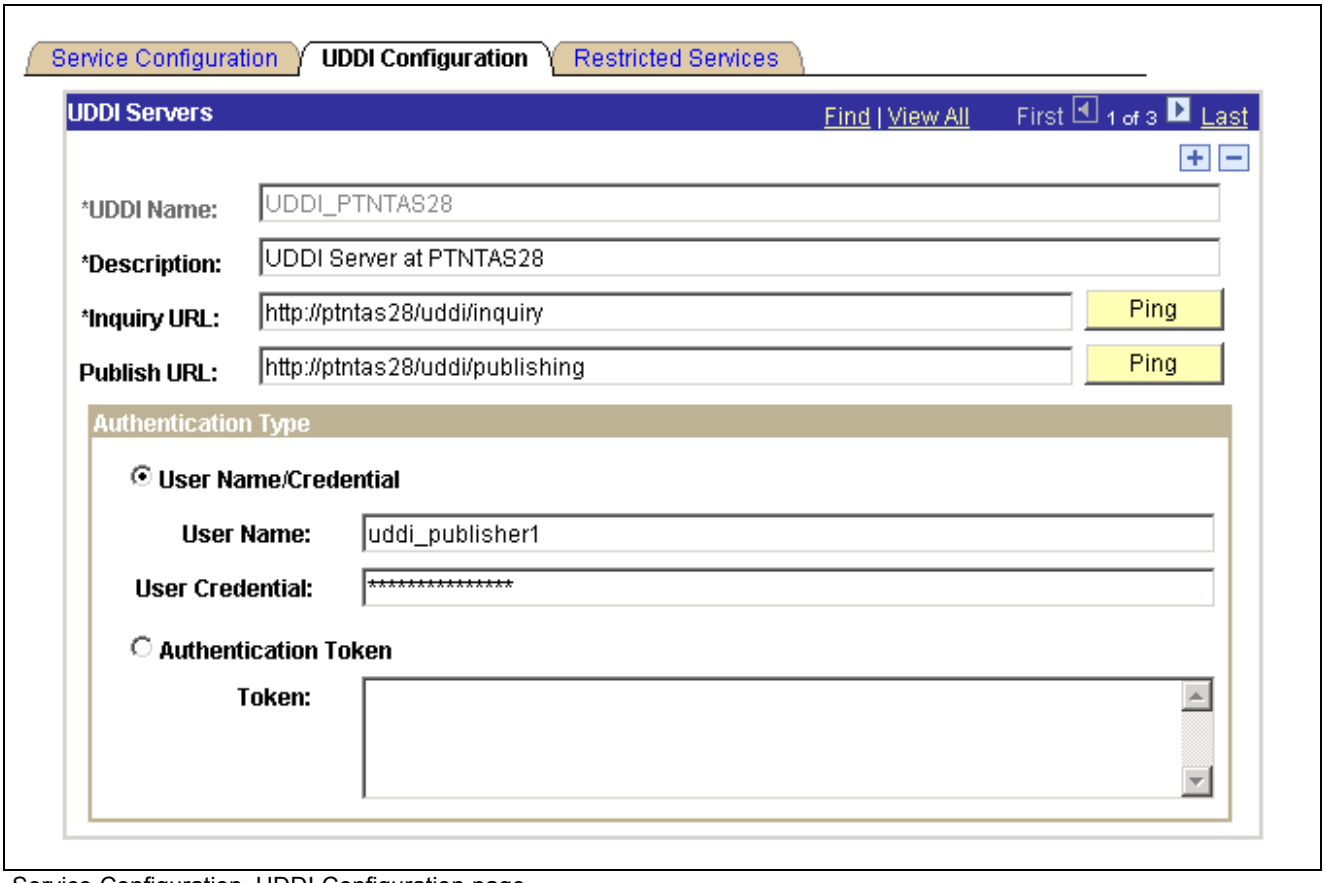
Page Used to Specify UDDI Repositories in the PeopleSoft System

Page Name	Object Name	Navigation	Usage
UDDI Configuration page	IB_SVCSETUP2	Select PeopleTools, Integration Broker, Configuration, Service Configuration. Click the UDDI Configuration tab.	Configure PeopleSoft Integration Broker to provide services to and consume services from UDDI repositories.

Specifying UDDI Repositories in the PeopleSoft System

Use the Service Configuration-UDDI Configuration page 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:



Service Configuration–UDDI Configuration page

To specify a UDDI repository in the PeopleSoft system:

1. Access the UDDI Configuration page.
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.

Copyright © 1988-2007, Oracle. All rights reserved.

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.

See Also

Chapter 24, "Consuming Services," page 533

Chapter 23, "Providing Services," page 507

Accessing and Viewing Service Definitions

This section discusses how to.

- Access service definitions.
- View WSDL documents generated for services.
- View service operation information services.
- View messages defined for services.

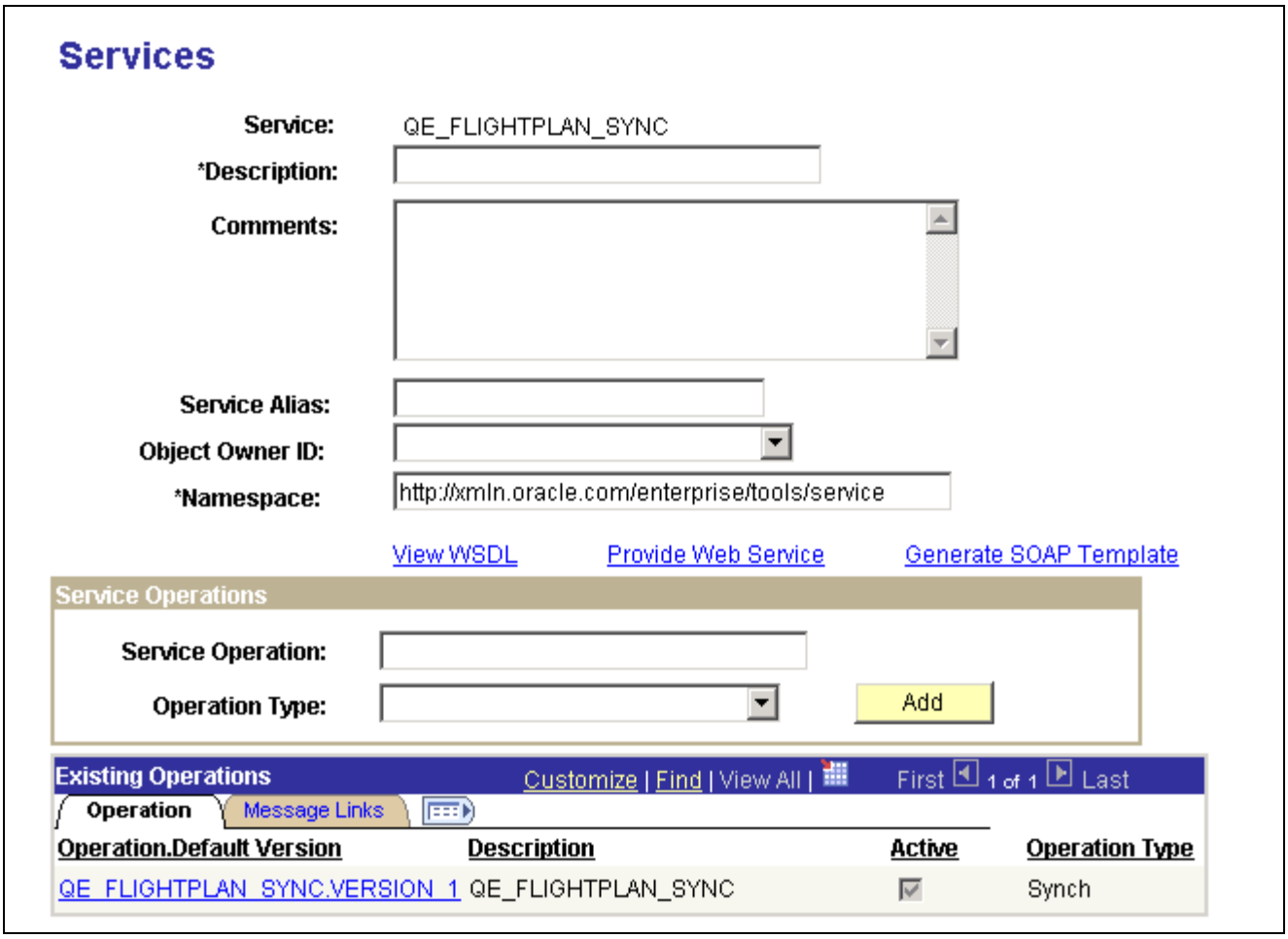
Pages Used to Access and View Service Definitions

Page Name	Object Name	Navigation	Usage
Services	IB_SERVICEDEFN	Select PeopleTools, Integration Broker, Integration Setup, Services.	Use this page to: • Access a service definition. • View the general information about the service.
WSDL Repository page	IB_SERVICEDEFN_SEC	From the Services page, click the View WSDL link.	Use this page to view WSDL generated for a service.
Services-General page	IB_SERVICE	From the Service page, click a hyperlinked service operations name.	View details about the default service operation version defined for a service.

Accessing Service Definitions

The following example shows the Services page:

 Copyright © 1988-2007, Oracle. All rights reserved.



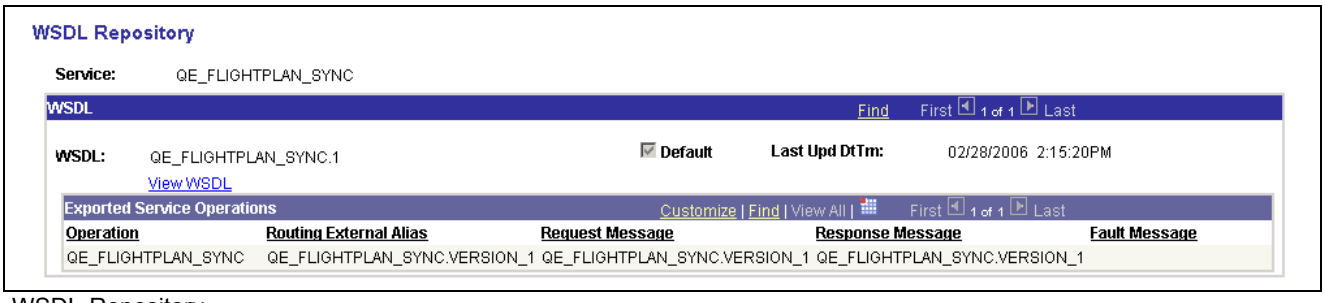
Services definition for the QE_FLIGHTPLAN_SYNC service

The top of the Service page displays general information about the service, including the name of the service, its description, its alias name, and so on.

Viewing WSDL Documents Generated for Services

Click the View WSDL link to display a page that provides a summary of all the WSDL documents that are generated for the service, as well as the service operations, request messages, response messages, and fault messages that are contained in each.

Note. Service operations must exist for a service to view WSDL documents for the service.



WSDL Repository

Click the View WSDL link to view the contents of the document.

Copyright © 1988-2007, Oracle. All rights reserved.

Click the Return button to return to the Services page.

Viewing Service Operation Information

The Existing Service Operations section of the Services page contains an Operation tab that displays service operations and service operation versions that are associated with the service. It also displays descriptions of the service operations, the type of operation, and whether the service operation is active.

When you click the name of a service operation, the operation opens on the Service Operations page, where you can view and modify service operations information, work with the service operation handlers, routing definitions, and do much more.

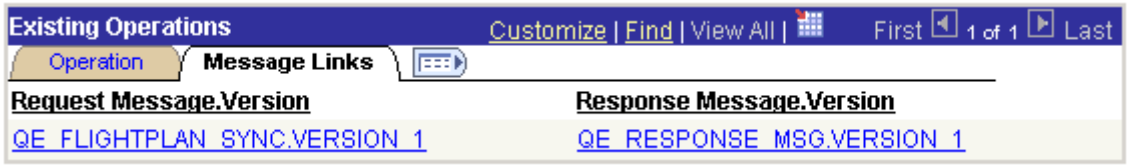
See Also

Chapter 15, “Managing Service Operations,” page 291

Viewing Messages Defined for Service Operations

The Existing Service Operations section of the Services page contains a Messages Links tab that displays the request and response messages defined for each service operation.

The following example shows the Message Links tab displaying request and response messages for the service operation that is associated with the QE_FLIGHTPLAN_SYNC service:



The request and response message for the QE_FLIGHTPLAN_SYNC service

Click the request or response message name to open the message in the Message Definitions page, where you can view and modify message definition information, message schema information, and more.

See Also

Chapter 10, “Managing Messages,” page 177

Adding Service Definitions

This section discuses how to add a service definition to the system

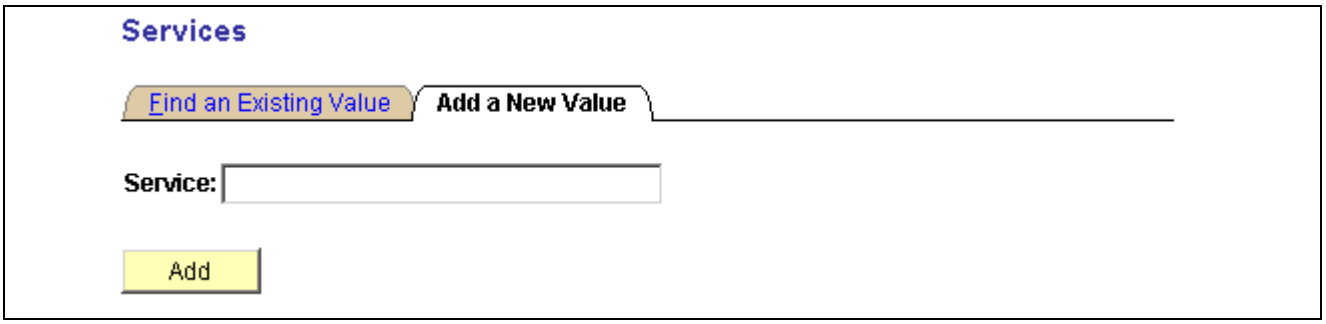
Page Used to Add Service Definitions

Page Name	Object Name	Navigation	Usage
Services	IB_SERVICEDEFN	Select PeopleTools, Integration Broker, Integration Setup, Services.	Add a service definition to the system.

 Copyright © 1988-2007, Oracle. All rights reserved.

Adding Services

To add a service definition to the system, use the Add a New Value tab on the Services search page. To access this page, select PeopleTools, Integration Broker, Integration Setup, Services. Then select the Add a New Value tab.



Add a New Value tab on the Services Search page

Note. Before you can add a service, you must configure PeopleSoft Integration Broker to handle services using the Service Configuration page.

To add a service definition:

1. Access the Services page and select the Add a New Value tab.
2. In the Service field, enter a name for the service.
3. Click the Add button.

The Services page appears, and you can now define the service.

See Also

Chapter 14, "Managing Services," Configuring Service Definitions, page 285

Chapter 14, "Managing Services," Setting Service Configuration Properties, page 279

Configuring Service Definitions

This section discusses how to configure service definitions.

Page Used to Configure Service Definitions

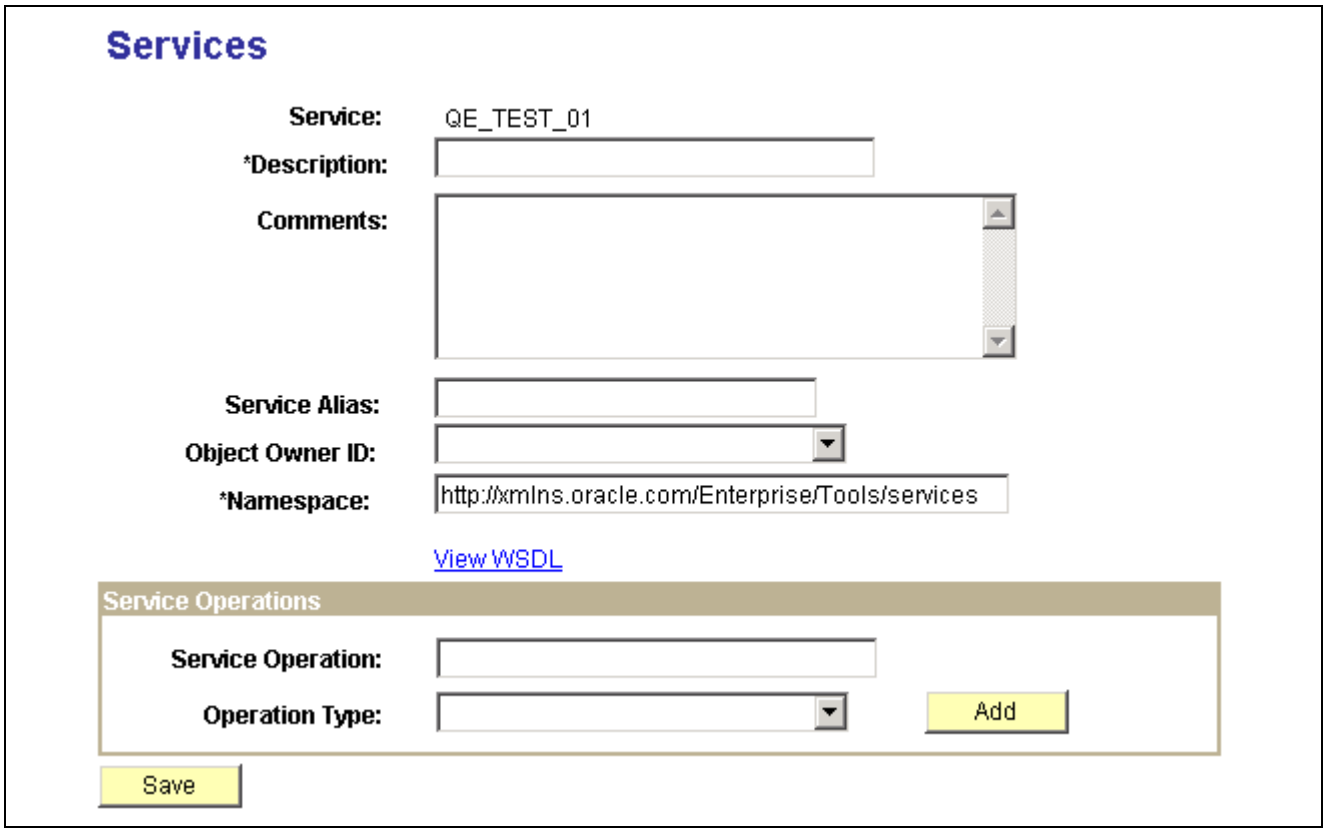
Page Name	Object Name	Navigation	Usage
Services	IB_SERVICEDEFN	Select PeopleTools, Integration Broker, Integration Setup, Services.	Add a service definition to the system.

Configuring a Service Definition

After you add a service definition to the system, the Services page appears and you can configure the service definition.

Copyright © 1988-2007, Oracle. All rights reserved.

To configure a service, use the Services page in the Services component (IB_SERVICEDEFN) in the PeopleSoft Pure Internet Architecture. The following example shows the Services page:



Configuring a service definition:

To configure a service definition:

1. If it is not already open, access the Services page.
2. In the Description field, enter a description for the service.
3. (Optional.) In the Comments field, enter comments about the service or the service.
4. (Optional.) In the Service Alias field, enter an alias name for the service.
5. (Optional.) From the Object Owner ID dropdown list box, select the owner of the service.
6. Enter a namespace URI for the service.

   The default value is the namespace that is declared in the Service Namespace field on the Service Configuration page.
7. Click the Save button.

You can add service operations to the service definition now or at a later time.

See Also

Chapter 14, “Managing Services,” Setting Service Configuration Properties, page 279

 Copyright © 1988-2007, Oracle. All rights reserved.

Restricting and Enabling Write Access to Services

This section provides an overview of restricting access to services and discusses how to:

- Restrict write access to services.
- Enable write access to services.

Understanding Restricting Write Access to Services

When you restrict write access to a service, sensitive fields of the service definition and of associated service operations appear in read-only mode. The following table lists the components and pages that contain fields and data that are related to services and describes the impact that restricting access to services has to each of them.

Restricted Component or Page	Restriction	Comments
Service	All fields are read-only.	NA
Service Operation	All fields are read-only, with the following exceptions: • User Password Required. • Non-Repudiation. • Runtime Schema Validation.	When a service is restricted, you cannot regenerate routings.
Handlers	All fields are read-only except. • The Status dropdown list box • The plus button that is used to add new handlers.	When a service is restricted, you can still activate or inactivate handlers.
Routings	All fields are read-only except: • The Inactivate Selecting Routings and Activate Selected Routings buttons. • The Add button.	When a service is restricted, you can: • Activate and deactivate routings of service operations that are associated with the service • Add new routings to service operations that are associated with the service.

You cannot delete or rename a restricted service. In addition, you cannot change, rename or delete any service operation that is defined as part of a restricted service.

Copyright © 1988-2007, Oracle. All rights reserved.

Page Used to Restrict and Enable Write Access to Services

Page Name	Object Name	Navigation	Usage
Service Configuration-Restricted Services	IB_SVCSETUP3	Select PeopleTools, Integration Broker, Service Configuration and click the Restrict Services tab.	Use the page to: • Restrict write access to services. • Enable write access to services

Restricting Write Access to Services

The following example shows the Restricted Services page.

Restricted Services page

To restrict write access to services:

1. Select PeopleTools, Integration Broker, Service Configuration. Click the Restricted Services tab. The Restricted Services page appears.
2. In the Service field, enter a service name and click the Search button, or click the Lookup button to search for a service.

   The service name or search results display in the Services list.
3. Check the Restricted Service check box next to the service name to which you want to restrict access.
4. Click the Save button.

Enabling Write Access to Services

To enable write access to services that you previously restricted:

1. Select PeopleTools, Integration Broker, Service Configuration. Click the Restricted Services tab.

   The Restricted Services page appears.
2. Select the service to write-enable using one of the following methods:
   - In the Service field, enter a service name and click the Search button.
   - Click the Lookup button to search for a service.
   - Check the Restricted Services check box, and click the Search button to display and select from all currently restricted services in the system.

Copyright © 1988-2007, Oracle. All rights reserved.

The service name or search results appear in the Services list.

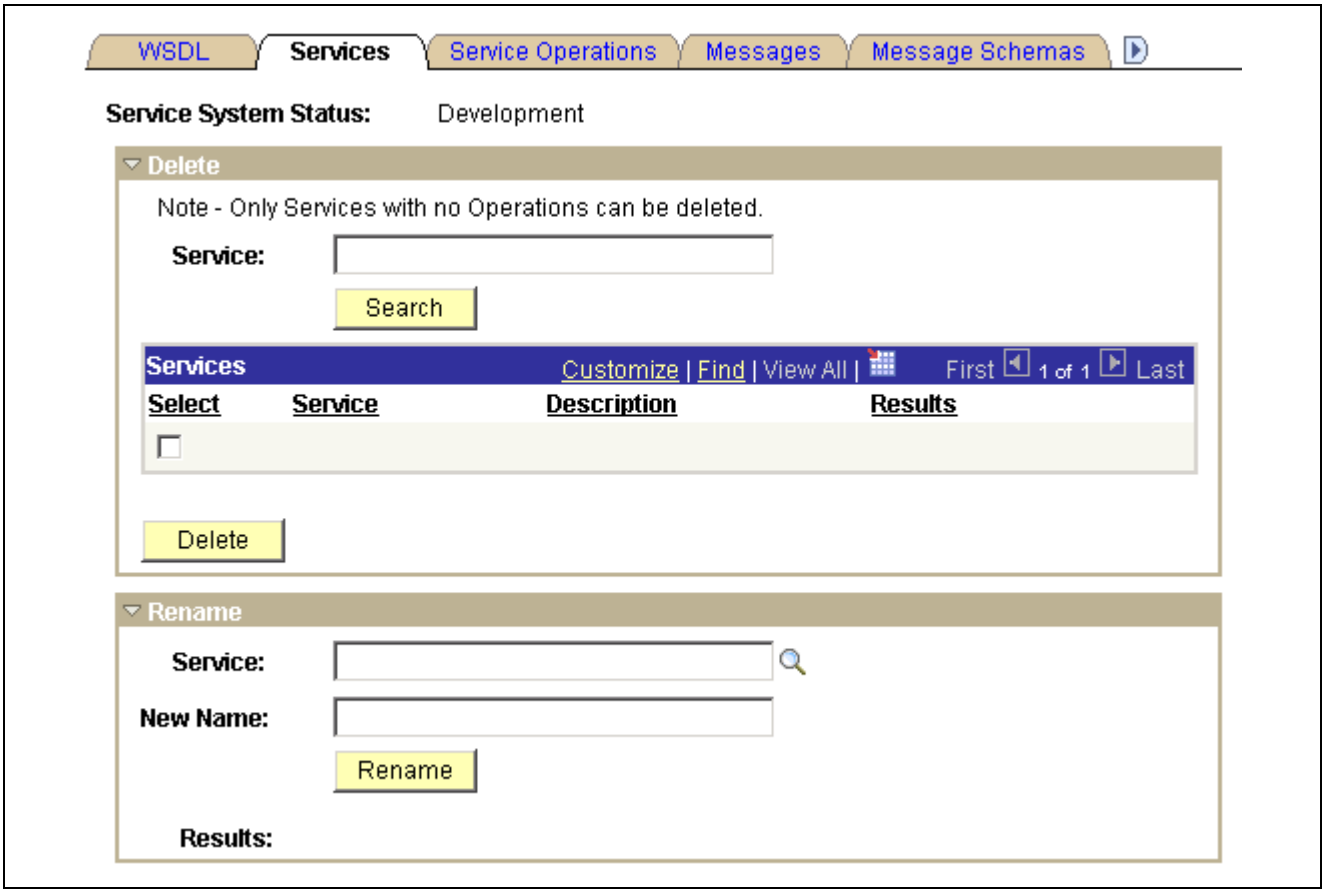
3. Clear the check box next to the service name to write access enable.
4. Click the Save button.

Renaming and Deleting Services

You can rename and delete services using the Services tab in the Service Administration component (IB_HOME_PAGE). The Services tab contains two sections: a Delete section that enables you to delete services and a Rename section that enables you to rename services.

When you first access the Services tab, both sections are collapsed. Click the section header arrow buttons to expand and collapse each section.

The following example shows the Services tab with both Delete and Rename sections expanded.



Services Administration – Services page with the Delete and Rename sections expanded

Copyright © 1988-2007, Oracle. All rights reserved.

Page Used to Rename and Delete Services

Page Name	Object Name	Navigation	Usage
Service Administration-Services page	IB_HOME_PAGE	Select PeopleTools, Integration Broker, Service Utilities. Select the Service tab.	Rename or delete services.

Renaming Services

The service system status that you set on the Services Configuration page affects the ability to rename services.

See Chapter 14, "Managing Services," Understanding Configuring PeopleSoft Integration Broker for Handling Services, page 277 and Chapter 14, "Managing Services," Setting Service Configuration Properties, page 279.

To rename a service:

1. Select PeopleTools, Integration Broker, Service Utilities. Click the Service tab.

   The Service page displays.
2. Click the arrow next to the Rename section header to expand the section.
3. In the Service field, enter the service to rename, or click the Lookup button to search for and select the service to rename.
4. In the New Name field, enter the new name for the service.
5. Click the Rename button.

After you click the Rename button, the Results field displays a message that the action was successful or displays a warning or error message with a description of the problem.

Deleting Services

You can delete services only when the service has no service operations associated with it. When you search for a service to delete, only such services that have no service operations associated with them are retrieved from the system.

To delete a service:

1. Select PeopleTools, Integration Broker, Service Utilities. Click the Service tab.

   The Service tab displays.
2. Click the arrow next to the Delete section header to expand the section.
3. In the Service field, enter the service name to delete, and click the Search button.

   Search results display in the results grid.
4. In the results grid, check the check box next to the service or services to delete.
5. Click the Delete button.

 Copyright © 1988-2007, Oracle. All rights reserved.

CHAPTER 15

Managing Service Operations

This chapter provides an overview of managing service operations and discusses how to:

- Access and view service operation definitions.
- Add service operation definitions.
- Define service operations.
- Set permissions to service operations.
- Change the services with which service operations are associated.
- Manage service operation versions.
- Attach files to service operations.
- Rename and delete service operations.

Understanding Managing Service Operations

This section discusses:

- Service operations.
- Service operation types.
- Service operation aliases.
- Service operation versions.
- Mapping service operations.

Service Operations

A *service operation* definition consists of general information about an operation, such as its name, description, and so on. It also specifies an operation type, which determines how the operation is to be processed, synchronously or asynchronously. In addition, it contains routings, which determine the direction, inbound or outbound, of the service operation. A service operation has one or more handlers, which contain and run the programming logic for sending or receiving the message, manipulating message content, and so on.

Note. Beginning with the PeopleTools 8.48 release, service operations house the processing logic found in messages, transactions and relationships used in earlier versions of PeopleSoft Integration Broker.

Service Operation Types

Service operation types determine the type of message processing. There are four service operation types:

Copyright © 1988-2007, Oracle. All rights reserved.

Asynchronous Request/Response	The sending system invokes a service operation asynchronously and processes the response from the receiving system asynchronously. Unlike a synchronous operation type, the response is not processed on the same thread as the response, and it is processed sometime in the future.
Asynchronous to Synchronous	The sending system's asynchronous process sends a synchronous request to a remote system. The sending asynchronous system expects the receiving system to send a synchronous response back. The sending asynchronous system transforms the response and puts it back in the queue for asynchronous consumption.
Asynchronous – One Way	The service operation is queued and sent in near real-time. Processing on the sending system continues without a response from the receiving system.
Synchronous	The service operation is provided in real-time. Processing on the sending system does not continue until it receives a response from the receiving system.

Service Operation Aliases

A *service operation alias* or *operation alias* is the service operation name that displays for the service operation when WSDL is provided.

Service operation aliases may be mixed case.

Duplicate service operation alias names within a service are not allowed.

Service Operation Versions

When you create a service operation, the operation that you create automatically becomes the default service operation version.

If you add a new version to the operation, the newly added version automatically becomes the default.

To continue to use a non-default service operation version, you must create a transformation.

Mapping Service Operations

You can map inbound, asynchronous, one-way service operations to more than one service operation.

Accessing and Viewing Service Operation Definitions

This section discusses how to:

- Access service operation definitions.
- View service operation definitions.

 Copyright © 1988-2007, Oracle. All rights reserved.

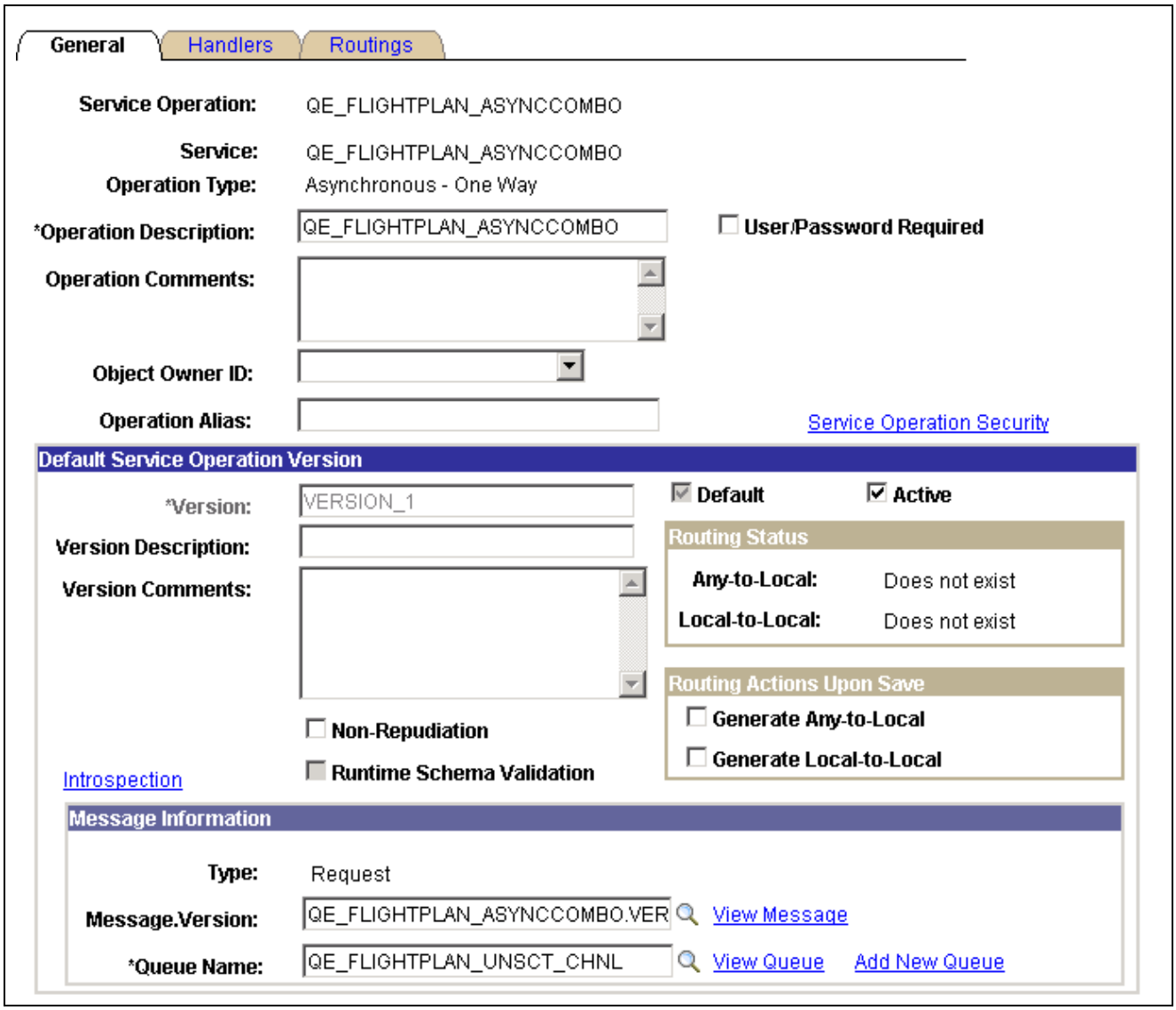
Pages Used to Access and View Service Operation Definitions

Page Name	Object Name	Navigation	Usage
Service Operations-General page	IB_SERVICE	Select PeopleTools, Integration Broker, Service Operations.	Use this page to: • Access a service operation definition. • View the following information about a service operation: - General information, such as name, description, service operation alias name, and so on. - Version information for the selected service operation. - Whether any-to-local or local-to-local routing definitions have been generated for the service operation. - Messages defined for the service operation.
Service Operation-Handlers page	IB_SERVICEHDLR	From the Service Operation-General page, click the Handlers tab.	View service operations handlers defined for the service operation.
Service Operation-Routings page	IB_SERVICERTNGS	From the Service Operation-General page, click the Routings tab.	View a list of routings defined for the service operation.

Accessing Service Operation Definitions

Use the pages in the Service Operations component (IB_SERVICE) to access and view service operation definitions. The following example shows the General tab of the Service Operations component:

Copyright © 1988-2007, Oracle. All rights reserved.



Service Operations – General tab

To access a service operation definition:

1. Select PeopleTools, Integration Broker, Integration Setup, Service Operations.

   The Find Service Operation search box appears.

2. Search for a service operation definition to view.

   You can search for an operation in one of two ways:

   • Click the Search button to display all service operation definitions in the system.

Copyright © 1988-2007, Oracle. All rights reserved.

- Enter search criteria in one or more of the following fields, and then click the Search button:

Service	Enter the service name that contains the service operation that you want to view, or click the Lookup button to search for and select a service name.
Service Operation	Enter the name of the service operation to view, or click the Lookup button to search for and select an operation.
Operation Type	From the Operation Type dropdown list box, select an operation type. Values are: • *Asynch Request/Response* • *Asynch to Synch* • *Asynchronous – One Way* • *Synchronous*

3. Click the name of the service operation to view.

   The Service Operations – General tab appears with data for the service operation that you selected.

Viewing Service Operation Definitions

The Service Operations component includes three pages:

General Tab	Features general-service and default-service operation information.
Handlers Tab	Provides summary information about handlers that have been added to an operation. Service operation handlers contain the programming logic for sending and receiving service operations and their contained messages, and for manipulating content.
Routings Tab	Provides summary information about service operations routings. Routing definition determine the direction—inbound, outbound or hub—of service operations.

Viewing General Service Operation Information

When you access a service operation, the General tab appears.

The top portion of this page contains basic information about a service operation, including its name, description, and so on. The Service Operation Security link opens the permission list for the service. Note that the Service Operation Security link appears only after a service operation definition is saved.

The Default Service Operation Version section displays service-operation version information, whether nonrepudiation is in effect, and whether runtime schema validation is enabled. The Introspection link enables you to access the Introspection and Validation page.

The Routing Status group box shows if any-to-local or local-to-local routing definitions have been generated for the service operation. Click the Routings tab to view detailed information about routing definitions that exist for a service operation.

The Routing Actions Upon Save group box shows the possible routings that the system can generate when the service operation definition is saved.

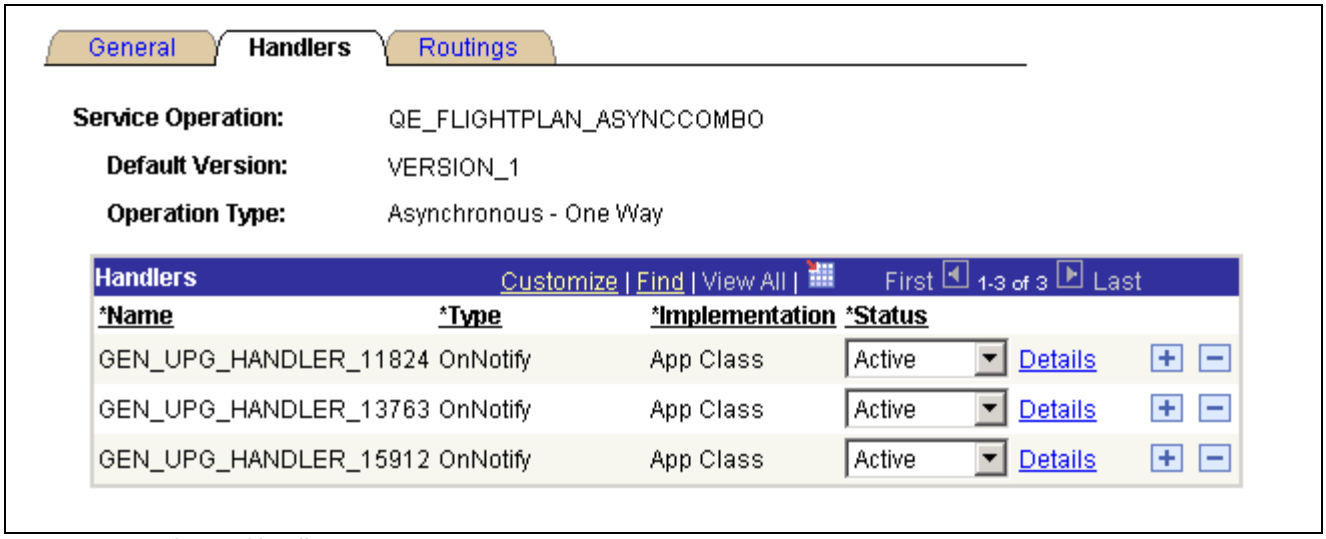
Copyright © 1988-2007, Oracle. All rights reserved.

The Message Information section displays the request message, response message information, and fault message for the service operation. The View Message links in this section open the displayed message on the Message Definition page, where you can view additional information about the message. For all operation types other than Synchronous, the queue to which a message belongs also appears. Click the View Queue link or the Add New Queue link to open the Queue Definition page to view additional queue definition information or to add or change a queue to which a message belongs.

See Chapter 18, "Managing Routing Definitions," Viewing Routing Definitions, page 335.

Viewing Handler Information

The Handlers tab of the Service Operation component lists summary information about handlers that have been added to an operation.

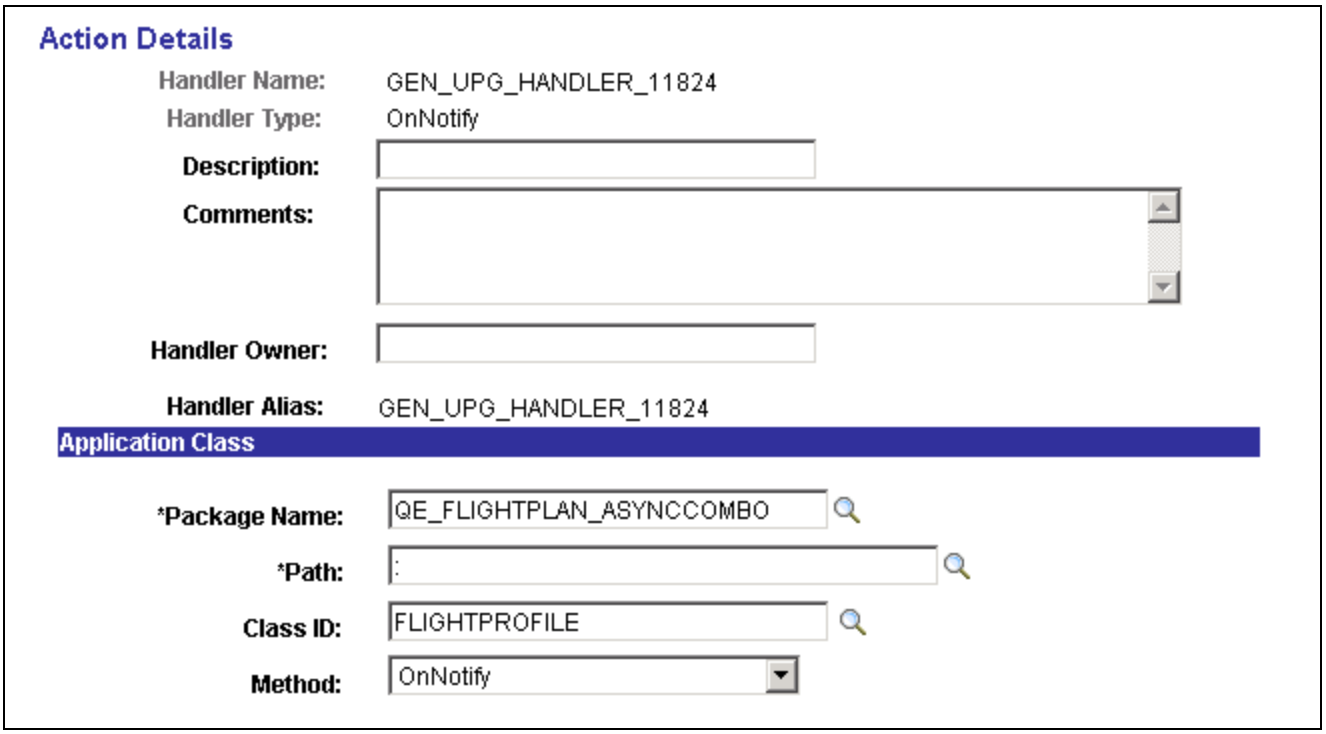


Service Operations – Handlers page

The summary information includes the handler name, the handler type, and the implementation method for the handler. The status of the handler, active or inactive, also appears.

Click the Details link to open the Action Details page for the handler. The following example shows the Actions Details page:

Copyright © 1988-2007, Oracle. All rights reserved.



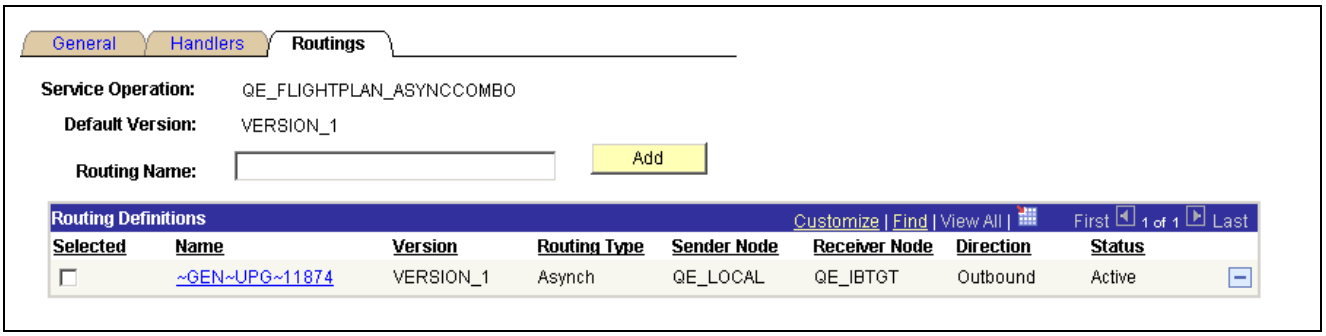
Handler Action Details page

The Action Details page shows additional information about the handler, including the owner and application class or component interface details.

You can also use this page to specify the handler details.

Viewing Routing Information

The Routing tab of the Service Operation component displays a summary of routing definitions for an operation.



Service Operations–Routings page

The Routings Definition grid on the page lists summary information for routings that are defined for a service operation. Summary information that is displayed includes the routing definition name, service operation version, routing type, sending node, receiving node, direction of the routing and the routing status.

Click a routing definition name to open the routing in the Routing Definitions component, where you can view additional information about the routing.

You can also use this page to add routing definitions to a service operation and to activate or deactivate routings for an operation.

Copyright © 1988-2007, Oracle. All rights reserved.

See Chapter 15, "Managing Service Operations," Adding Routing Definitions, page 303 and Chapter 15, "Managing Service Operations," Activating and Inactivating Routing Definitions, page 304.

Adding Service Operation Definitions

This section discusses how to add a service operation definition.

Page Used to Add Service Operation Definitions

Page Name	Object Name	Navigation	Usage
Service Operation-General page	IB_SERVICE	Select PeopleTools, Integration Broker, Service Operations.	Add a service operation definition.

Adding a Service Operation Definition

To add a service operation definition:

1. Select PeopleTools, Integration Broker, Integration Setup, Service Operations.

   A search page appears.
2. Click the Add Service Operation tab.
3. In the Service field, enter the service name to which the new operation will belong or click the Lookup button to search for a service name.
4. In the Service Operation field, enter a name for the service operation.
5. From the Operation Type dropdown list box, select an operation type. Values are:
   - *Asynchronous – One Way*
   - *Synchronous*
   - *Asynch Request/Response*
   - *Asynch to Synch*
6. Click the Add button.

The new definition appears on the General tab of the Service Operation component, and you can now define the service operation.

Configuring Service Operation Definitions

After you add a service operation definition to the system, you can define the service operation.

This section discusses how to:

- Specify general information.
- Define service-operations version information.

 Copyright © 1988-2007, Oracle. All rights reserved.

- Add handlers to service operations.
- Add routing definitions.
- Activate and inactivate routings.

Pages Used to Configure Service Operation Definitions

Page Name	Object Name	Navigation	Usage
Service Operations-General page	IB_SERVICE	Select PeopleTools, Integration Broker, Service Operations.	Use this page to define: • General service operation information, such as name, description, service operation alias name, and so on. • Version information for the selected service operation. • Specify messages for the service operation. • Generate local-to-local and any-to-local routing definitions.
Service Operation-Handlers page	IB_SERVICEHDLR	From the Service Operation-General page, click the Handlers tab.	Define handlers for a service operation.
Service Operation-Routings page	IB_SERVICERTNGS	From the Service Operation-General page, click the Routings tab.	Define, activate and inactivate point-to-point routing definitions.

Specifying General Information

To specify general information:

1. Access the Service Operations-General page.
2. In the Operation Description field, enter a description for the operation.
3. (Optional.) Check the User ID/ Password Required check box to require a user ID and password for inbound service operations.

   See Chapter 27, “Setting Up Secure Integration Environments,” Managing User Authentication, page 637.
4. (Optional.) In the Operation Comments field, enter comments about the service operation .
5. (Optional.) From the Object Owner ID field, select the owner of the definition.

   The owner ID helps to determine the application team that last made a change to a service definition. The values in the dropdown list box are translate table values that you can define in the OBJECTOWNERID field record.
6. (Optional.) In the Operation Alias field, enter an alias name for the service operation.

The general information section of this page includes a Service Operation Security link. Granting permissions to service operations is discussed elsewhere in this chapter.

Copyright © 1988-2007, Oracle. All rights reserved.

Before you can save the service operation definition, you specify messages for the service operation, as described in the next section.

See Chapter 15, "Managing Service Operations," Setting Permissions to Service Operations, page 304.

The following section continues to describe how to define a service operation and discusses how to assign default versions to service operations.

Defining Service Operation Version Information

When you first create a service operation definition, the definition that you initially define is the default version.

When the newly created service operation definition opens, the Default check box is enabled and is read-only.

You can subsequently define additional service operation versions and assign them as the default.

See Chapter 15, "Managing Service Operations," Managing Service Operation Versions, page 306.

Defining General Version Information

To define the service operation default version:

1. In the Version field, enter a version identifier.

   The default is *v1*.

2. (Optional.) In the Version Description field, enter a description for the operation version.

   If you enter no information, the description by default is the name of the service operation when you save the definition.

3. (Optional.) In the Version Comments box, enter comments about the version.
4. (Optional.) Check the Non-Repudiation check box to apply nonrepudiation to the message.
5. (Optional.) Check the Runtime Schema Validation check box to enable service schema validation at runtime.

You can click the Introspection link to employ introspection to generate point-to-point routings.

See Chapter 18, "Managing Routing Definitions," Using Introspection to Create Routing Definitions, page 347.

Continue to the next section to specify messages for service operations. You cannot save the service operation definition until you define messages for it.

Specifying Messages for Service Operations

You specify messages for service operations in the Message Information section of the Service Operations – General page.

The messages that you specify define the structure of the data that is contained in the service operation.

The service operation type determines the number of messages and message types (request or response) that you specify.

To specify messages for a service operation:

1. Locate the Message Information section on the Service Operations – General page.
2. Locate the Type field, and take note of the message type to define.
3. In the Message.Version field, enter the message name followed by a dot and version, or click the Lookup button to search for one.

 Copyright © 1988-2007, Oracle. All rights reserved.

After you select the message, you can click the View Message link to view the message.

4. Specify the queue for the message.

   Note. If you are defining a message for a synchronous operation type, you do not need to define a queue.

   Your options are:

   - In the Queue Name field, enter the queue name.
   - Click the Lookup button to search for a queue.
   - Click the Add Queue link to open the Queue Definitions page and define a new queue for the message.

     See Chapter 11, "Managing Service Operation Queues," Adding Queue Definitions, page 197.

5. Repeat steps 1 through 4 for each message type that appears in the Message Information section.
6. Click the Save button.

Specifying Fault Messages for Service Operations

You can specify fault messages for service operations for error handling.

Note. You cannot add fault messages to asynchronous service operations.

To specify a fault message:

1. Locate the Default Service Operation Version section on the Service Operations – General tab.
2. Click the Add Fault Type button.

   A new row appears in which to specify a message. Note that the Type field in the new row displays *Fault.*
3. In the Message.Version field, enter the message name, or click the Lookup button to search for one.

   After you select the message, you can click the View Message link to view the message.
4. Click the Save button.

To delete a fault message, in the Default Service Operation Version section, click the Delete Fault Type button. Then click the Save button.

Generating Local-to-Local and Any-to-Local Routing Definitions

Use the Service Operations-General page to initiate generating local-to-local and any-to-local routing definitions.

See Chapter 18, "Managing Routing Definitions," Initiating System-Generated Routing Definitions, page 336.

Adding Handlers to Service Operations

This section discusses how to:

- Add handlers.
- Add handler details.
- Add handler implementation details.

Copyright © 1988-2007, Oracle. All rights reserved.

Adding Handlers

To add handlers to a service operation:

1. On the Service Operations component, click the Handlers tab.

   The Handlers page appears.

2. In the Handlers section, enter a handler name in the Name field.

   Note that for OnRequest, and OnRoute handlers, you need not enter a name. The system adds a handler name after you provide the handler details.

3. From the Type dropdown list box, select the handler type.

   The service operation type determines the handler types that are available to choose.

4. From the Implementation dropdown list box, select the method to use to implement the handler.

   The service operation type determines the handler types that are available to choose.

5. From the Status dropdown list box, select a status for the handler. Values are:
   - *Active.* Select to make the handler active.
   - *Inactive.* Select to make the handler inactive.

Continue to the next section for information about entering handler details.

Adding Handler Details

This section describes how to add handler details.

Note. You do not enter handler details for handler implementations using data mover scripts or for the deprecated PeopleCode handler.

To add handler details:

1. On the Handlers tab of the Service Operations – Handlers tab, in the Handlers section, click the Details link.

   The Action Details page appears.

2. (Optional.) In the Description field, enter a description for the handler.
3. (Optional.) In the Comments field, enter comments about the handler.
4. (Optional.) In the Handler Owner field, enter the name of the person or group that owns or maintains the handler.

Continue to the next section for information about adding handler implementation details

Adding Handler Implementation Details

This section describes how to add handler implementation details.

Note. You do not enter handler implementation details for handler implementations using data mover scripts or for the deprecated PeopleCode handler.

To add handler implementation details:

1. On the Service Operations – Handlers tab, in the Handlers section, click the Details link.

 Copyright © 1988-2007, Oracle. All rights reserved.

The Action Details page appears.

2. Enter details based on the implementation method that is selected:

Application Class

When the implementation method is an application class, complete the following fields:

- Package Name

  Enter the package name that contains the class that you want to specify, or use the Lookup button to search for and select one.

- Path

  Enter the name or names of any subpackages that contain the application class that you want to specify, or use the Lookup button to search for and select one.

- Class ID

  Enter the name of the application class that contains the method that you want to specify, or use the Lookup button to search for and select one.

  Only application classes that implement an appropriate base class are shown. The base class is dependent on the handler type.

- Method

  From the Method dropdown list box, select the method from the selected application class that you want to specify.

  Only methods with the correct signature for the current handler type are shown.

Component Interface

When the implementation method is a component interface, complete the following fields:

- CI Name

  Enter the component interface name, or use the Lookup button to search for and select one.

- Method

  From the Method dropdown list box, select a method

3. Click the OK button.
4. Click the Save button.

Adding Routing Definitions

This section describes how to create point-to-point service operation routing definitions from the Service Operations – Routing page.

Note. You can also create routings using the Routings component, the Introspection component, or the Routings page in the Node Definitions component.

To add a routing to a service operation:

1. On the Service Operations component, click the Routings tab.

Copyright © 1988-2007, Oracle. All rights reserved.

The Routings page appears.

2. In the Routing Name field, enter a name for the routing.
3. Click the Add button.

   The Routing Definition page appears.

   Creating and defining a routing is discussed elsewhere in this PeopleBook.

   See Chapter 18, "Managing Routing Definitions," Creating Routing Definitions, page 338.

The next section describes how to activate routings.

Activating and Inactivating Routing Definitions

To activate or inactivate a routing:

1. On the Service Operations component, click the Routings tab.

   The Routings page appears.
2. Check the box in the Select column next to the routing definition names that you want to activate or inactivate.
3. Activate or inactivate the routing definition.
   - To activate the routings, click the Activate Selected Routings button.
   - To inactivate the routings, click the Inactivate Selected Routings button.
4. Click the Save button.

Setting Permissions to Service Operations

This section describes how to use the Service Operations component to set permissions to access service operations. You can also set these permission in the Security component.

Understanding Setting Permission to Service Operations

Security operations are secured using permission lists.

When you select the User/Password Required check box on the Service Operations-General page, on inbound integrations, your integration partners must supply a valid user ID that is associated with the permission list you specify to invoke service operations.

Page Used to Set Permissions to Service Operations

Page Name	Object Name	Navigation	Usage
Web Service Access page	WS_ACCESS_IB	From the Service Operations-General page, click the Service Operation Security link.	Set permissions to service operations.

 Copyright © 1988-2007, Oracle. All rights reserved.

Setting Permission Access to Service Operations

To grant permissions to service operations:

1. Select PeopleTools, Integration Broker, Integration Setup, Service Operations, and select a service operation with which to work.

   The Service Operations - General page appears.

2. Click the Service Operation Security link.

   The Web Service Access page appears.

3. In the Permission List field, enter a permission list for the service operation, or click the Lookup button to search for and select one.

4. From the Access dropdown list, select an access level for the service operation. Values are:

   - *Full Access.* (Default.)
   - *No Access.*

5. Click the Save button.

See Also

Enterprise PeopleTools 8.49 PeopleBook: Security Administration, "Setting Up Permission Lists"

Changing the Services with Which Service Operations are Associated

This section discusses how to change the services with which service operations are associated.

Page Used to Change the Services with Which Service Operations are Associated

Page Name	Object Name	Navigation	Usage
Service Administration-Service Operations	IB_HOME_PAGE2	Select PeopleTools, Integration Broker, Service Utilities, Services Administration and click the Service Operations tab	Change the service with which a service operation is associated.

Changing the Service with Which a Service Operation is Associated

Use the Services Operations page in the Services Administration component to change the services with which service operations are associated. The Services Operations page contains three sections: a Delete section that enables you to delete service operations, a Rename section that enables you to rename service operations, and a Service Change section that enables you to change the service with which a service operation is associated.

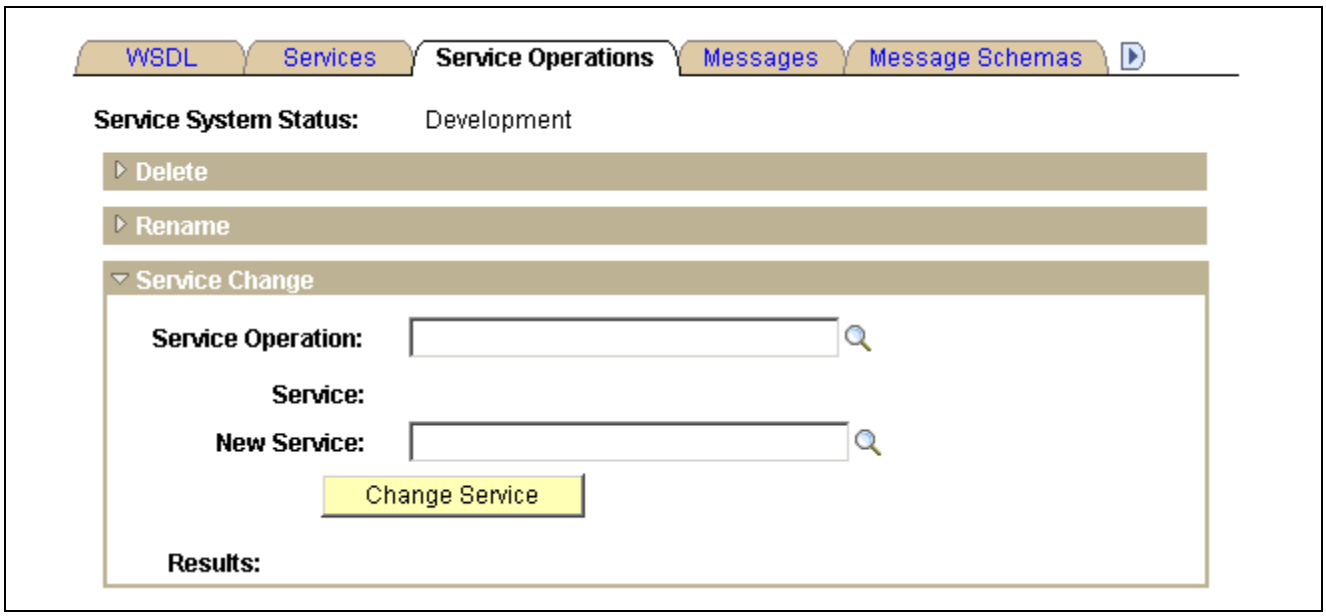
Using this page to rename and delete service operations is discussed elsewhere in this chapter.

See Chapter 15, "Managing Service Operations," Renaming and Deleting Service Operations, page 311.

Copyright © 1988-2007, Oracle. All rights reserved.

When you first access the Services Operations page, all sections are collapsed. Click the section header arrow buttons to expand and collapse each section.

The following example shows the Services Administration – Service Operations page with the Change Service section expanded:



Services Administration – Service Operations page with the Service Change section expanded

The service system status that is set on the Service Configuration page affects the ability to change the services that are associated with service operations.

The service system status that you set on the Services Configuration page affects the ability to rename service operations.

See Chapter 14, "Managing Services," Understanding Configuring PeopleSoft Integration Broker for Handling Services, page 277 and Chapter 14, "Managing Services," Setting Service Configuration Properties, page 279.

To change the service with which a service operation is associated:

1. Access the Services Administration – Service Operations page.
2. In the Service Operation field, enter a service operation name or use the Lookup button to search for and select one.

   When you select a service operation, the service to which it is currently associated appears in the Service field.
3. In the New Service field, enter the name of the service to associate with the service operation, or click the Lookup button to search for and select one.
4. Click the Change Service button.

Managing Service Operation Versions

This section discusses how to:

- Create service operation versions.

 Copyright © 1988-2007, Oracle. All rights reserved.

- Use non-default service operation versions.

Page Used to Manager Service Operation Versions

Page Name	Object Name	Navigation	Usage
Service Operation Versions page	IB_SERVICEVERS	From the Service Operations-General page, click the Add Version link.	Add a new service operation version.

Creating Service Operation Versions

When you create a new service operation version, the new version automatically becomes the active default version.

If you have generated WSDL for the current service operation, after you create the new version you are prompted to generate WSDL for the new version.

To create a new service operation version:

1. Select PeopleTools, Integration Broker, Integration Setup, Service Operations.

   Select the service operation with which to work. The Service Operations–General page appears.
2. At the bottom of the page, click the Add Version hyperlink.

   The Service Operation Versions page appears.
3. In the Service Operations Version field, enter the new version and click the OK button.

   The Service Operations Version page appears.
4. Complete the fields as appropriate for the new service operation version.

   See Chapter 15, "Managing Service Operations," Defining Service Operation Version Information, page 300.
5. Click the Save button.

The new service operation version appears in the Service Operations–General page.

A Non-Default Versions grid appears at the bottom of the page that lists and provides access to the previous service operation version. Note that all versions that display in this grid have a status of *Inactive*.

Using Non-Default Service Operation Versions

To continue using non-default service operation version you must write and apply transformations to transform message shapes contained in the non-default service operation version to the message shapes contained in the default version.

You need to write and apply transformations only for the changed message shapes. For example, if a service operation contains request and response messages, but only the request message shape has changed between versions, you need only write and apply a transform program to transform the request message on the request message that is contained in the non-default service operation version to the shape of the request message in the default version.

The non-default versions are inactive until the transformations are entered and the status is changed to Active. Then the grid shows the version with Active.

Copyright © 1988-2007, Oracle. All rights reserved.

Attaching Files to Service Operations

This section provides an overview of attaching files to service operations and discusses how to:

- Use the FTP Attachment utility.
- Send attachment information with service operations.
- Process attachment information that is included in service operations.

Understanding Attaching Files to Service Operations

PeopleSoft Integration Broker provides an FTP Attachment Upload utility that enables you to upload attachments from any directory to your FTP server and then provide the location of the attachments in service operation PeopleCode. The attachments can be in any file format, such as text files, EDI files, word processing files, and so on.

As the operation is consumed, you can access these attachments using the attachment API. The recipient can get the necessary information about the attachment and can retrieve it using a URL or file location that you provide.

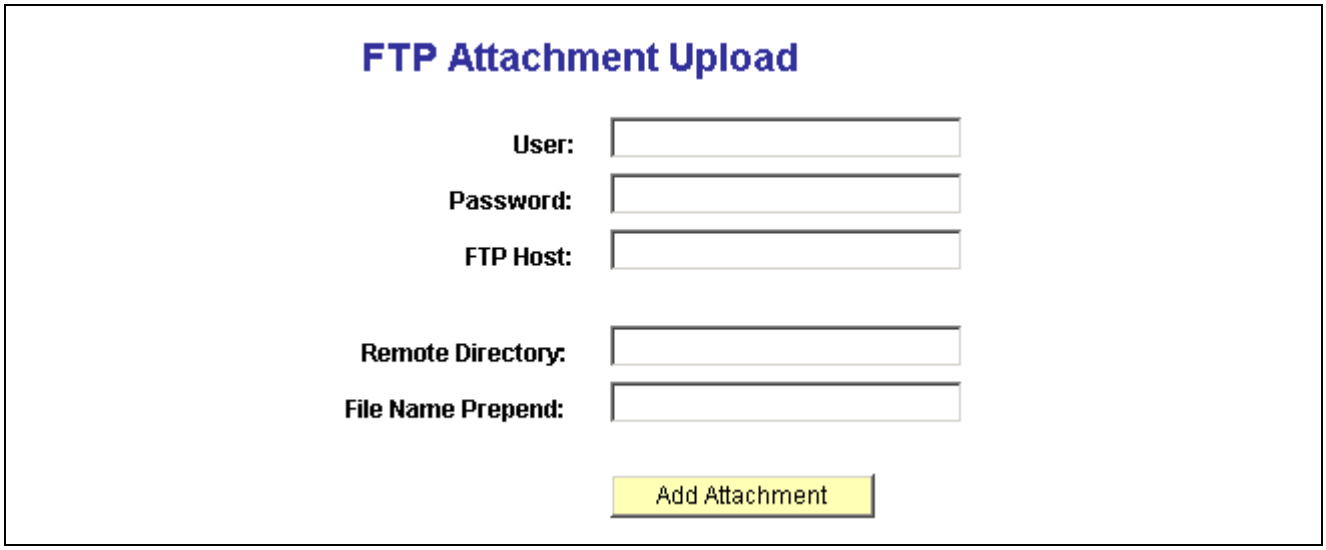
Note. The FTP Attachment Upload utility does not support uploading attachments from the database. To upload attachments from the database, manually retrieve and copy them to the FTP server.

Page Used to Attach Files to Service Operations

Page Name	Object Name	Navigation	Usage
FTP Attachment Upload	IB_FILEUPLOAD	Select PeopleTools, Integration Broker, Files Utilities, File Upload	Upload files to your FTP server.

Using the FTP Attachment Utility

Use the FTP Attachment Upload page in the Files Utilities component (IB_FILEUPLOAD) to upload files to your FTP server for attaching to service operations. The page is shown in the following example:



FTP Attachment Upload page

Copyright © 1988-2007, Oracle. All rights reserved.

You work with the following page elements:

User	Indicates the user ID of the FTP server.
Password	Indicates the password to the FTP server.
FTP Host	Indicates the machine name of the FTP server.
Remote Directory	Indicates the directory path to the file to upload.
File Name Prepend	Enter text to prepend the file name to build the final file name to copy to the target directory.
Add Attachment	Click to upload the indicated file.

Sending Attachment Information with Service Operations

The following example shows sample PeopleCode for sending attachment information:

```
Local Message &MSG;
Local Rowset &Flight_Profile;
Local String &Attachment_id;

QE_FLIGHTDATA.QE_ACNUMBER.Value = QE_FLIGHTDATA.QE_ACNUMBER + 1;

&FLIGHT_PROFILE = GetLevel0();

&MSG = CreateMessage (Operation.ASYNC_RR);

&Attachment_id = &MSG.IBInfo.AddAttachment (c:\\temp\\myfile.txt);

&attachReturn = &MSG.IBInfo.SetAttachmentProperty(&Attachment_id,
     %Attachment_Encoding, "UTF-8");

&attachReturn = &MSG.IBInfo.SetAttachmentProperty(&Attachment_id,
     %Attachment_Base, "Standard");

&attachReturn = &MSG.IBInfo.SetAttachmentProperty(&Attachment_id,
     %Attachment_Disposition, "Pending");

&attachReturn = &MSG.IBInfo.SetAttachmentProperty(&Attachment_id,
     %Attachment_Language, "English");

&attachReturn = &MSG.IBInfo.SetAttachmentProperty(&Attachment_id,
     %Attachment_Description, "Parts data");

&MSG.CopyRowset (&FLIGHT_PROFILE);

%IntBroker.Publish(&MSG);
```

Copyright © 1988-2007, Oracle. All rights reserved.

Processing Attachment Information Included in Service Operations

The following example shows sample PeopleCode for processing an attachment from a notification:

```
import PS_PT:Integration:INotificationHandler;

class FLIGHTPROFILE implements PS_PT:Integration:INotificationHandler;
     method FLIGHTPROFILE();
     method OnNotify(&MSG As Message);

end-class;

/* Constructor */
method FLIGHTPROFILE
     %Super = create PS_PT:Integration:INotificationHandler();
end method;

method OnNotify
     /+ $MSG as Message +/
     /+ Extends/implements PS_PT:Integration:INotificationHandler.OnNotify +/

     Local Rowset &rs;
     Local integer &count;
     Local string &Attachment_ID &Results;

     &rs = &MSG.GetRowset();

     &count = &MSG.IBInfo.NumberOfAttachments;
     If &count > 0 Then

          &Attachment_ID = &MSG.IBInfo.GetAttachmentContentID(1);
          &Results = &MSG.IBInfo.GetAttachmentProperty(&Attachment_ID,
            %Attachment_Encoding);
          &Results = &MSG.IBInfo.GetAttachmentProperty(&Attachment_ID,
            %Attachment_Type);
          &Results = &MSG.IBInfo.GetAttachmentProperty(&Attachment_ID,
            %Attachment_URL);
          &Results = &MSG.IBInfo.GetAttachmentProperty(&Attachment_ID,
            %Attachment_Base);
          &Results = &MSG.IBInfo.GetAttachmentProperty(&Attachment_ID,
            %Attachment_Location);
          &Results = &MSG.IBInfo.GetAttachmentProperty(&Attachment_ID,
            %Attachment_Disposition);
          &Results = &MSG.IBInfo.GetAttachmentProperty(&Attachment_ID,
             %Attachment_Description);

     End-If;
     /* Process data from message */
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
end-method;
```

Renaming and Deleting Service Operations

You can rename and delete service operations using the Services Operations page in the Service Administration component. The Services Operations page contains three sections: a Delete section that enables you to delete service operations, a Rename section that enables you to rename service operations, and a Service Change section that enables you to change the service with which a service operation is associated.

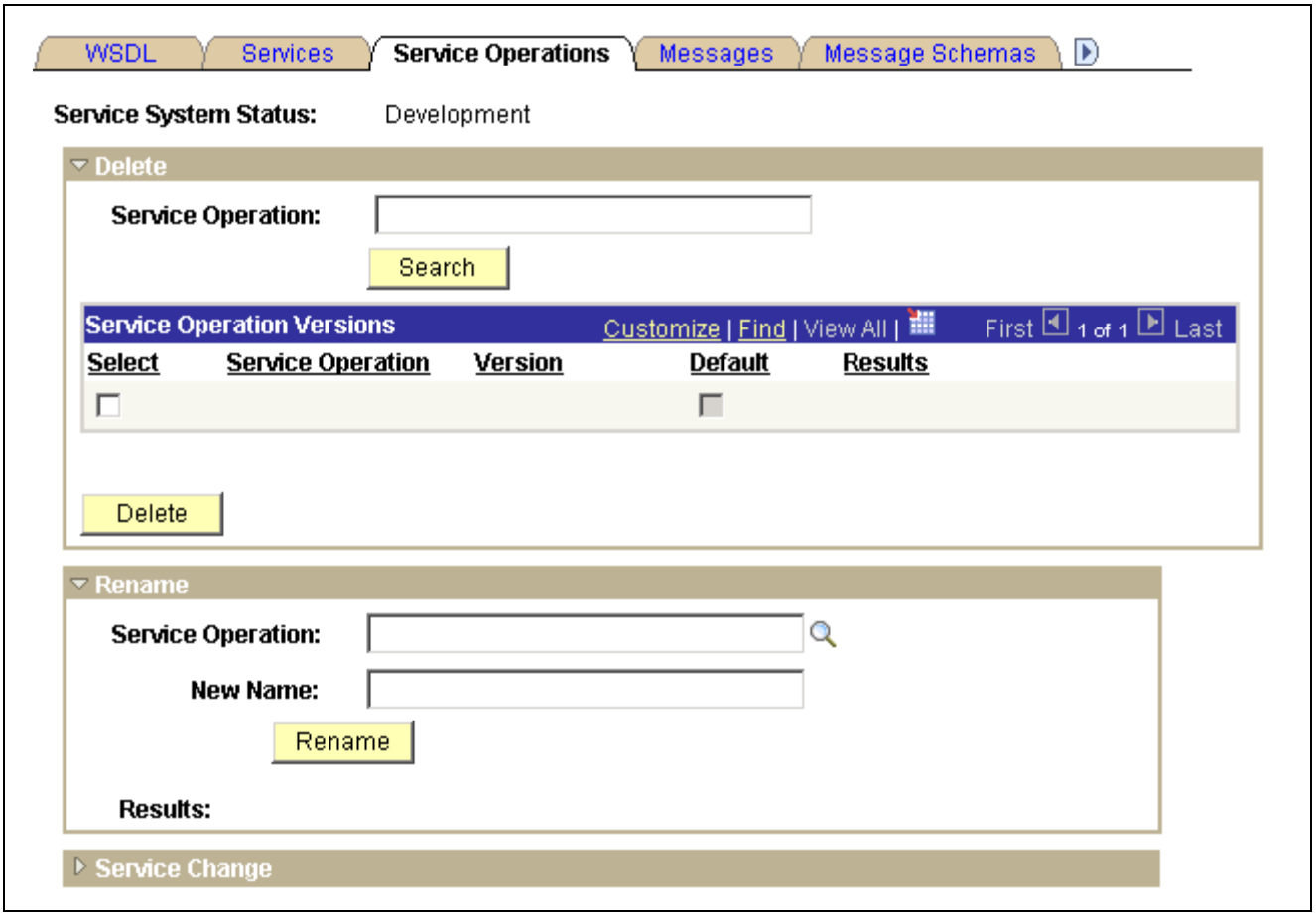
Changing the service with which a service operation is associated is discussed elsewhere in this chapter.

See Chapter 15, "Managing Service Operations," Changing the Services with Which Service Operations are Associated, page 305.

To access the page, select PeopleTools, Integration Broker, Service Utilities, Services Administration and click the Service Operations tab.

When you first access the Services Operations page, all sections are collapsed. Click the section header arrow buttons to expand and collapse each section.

The following example shows the Services Operation page with the Delete and Rename sections expanded:



Services Administration – Service Operations page with the Delete and Rename sections expanded

Copyright © 1988-2007, Oracle. All rights reserved.

At the top of the page, the Service System Status displays the current setting. The service system status, set in the Services Configuration page, impacts the ability to rename and delete messages.

See Chapter 14, "Managing Services," Understanding Configuring PeopleSoft Integration Broker for Handling Services, page 277 and Chapter 14, "Managing Services," Setting Service Configuration Properties, page 279.

Page Used to Rename and Delete Service Operations

Page Name	Object Name	Navigation	Usage
Service Administration-Service Operations page	IB_HOME_PAGE2	Select PeopleTools, Integration Broker, Service Utilities, Services Administration and click the Service Operations tab.	Rename and delete service operations.

Renaming Service Operations

Renaming a service operation is allowed only if the operation is not referenced in any runtime table. If a service operation is referenced in a runtime table, you must archive the data before you can rename the operation.

You cannot rename service operations associated with the restricted services *IB_UTILITY* and *IB_GENERIC* .

To rename a service operation:

1. Access the Services Administration – Service Operations page.
2. Click the arrow next to the Rename section header to expand the section.
3. In the Service Operation field, enter the service to rename, or click the Lookup button to search for and select the service operation to rename.
4. In the New Name field, enter the new name for the service.
5. Click the Rename button.

Deleting Service Operations

You can delete service operations individually, with the exception of the default service operation version. If you mark the default service operation for deletion, the system marks all versions for deletion and the entire service operation is deleted.

Note. When you delete a service operation, you are actually deleting a version of a service operation. An service operation cannot exist without at least one default version.

Deleting a service operation is allowed only if the operation is not referenced in any runtime table. If a service operation is referenced in a runtime table, you must first archive the data before deleting the service operation. Use the Service Operations Monitor to archive data.

See Chapter 21, "Using the Service Operations Monitor," Archiving Service Operation Instances, page 453.

You cannot delete service operations associated with the restricted services *IB_UTILITY* and *IB_GENERIC* .

To delete a service operation:

1. Access the Services Administration – Service Operations page.
2. Click the arrow next to the Delete section header to expand the section.

 Copyright © 1988-2007, Oracle. All rights reserved.

3. In the Service Operations field, enter the service operation name to delete and click the Search button. Search results appear in the results grid.
4. In the results grid, check the box next to the service operation or service operations to delete.
5. Click the Delete button.

Copyright © 1988-2007, Oracle. All rights reserved.

 Copyright © 1988-2007, Oracle. All rights reserved.

CHAPTER 16

Enabling Runtime Message Schema Validation

This chapter discusses how to:

- Select service operations.
- View defined message schemas.
- Enable runtime message schema validation.

Understanding Message Schema Validation

PeopleSoft Integration Broker enables you to validate, at runtime, the messages defined in service operations against message schemas.

Message Schema Validation

Validating messages against message schemas can ensure that during integration development, no changes or deletions were inadvertently made to the message. You can use schema validation on outbound and inbound messages, provided that you enable schema validation.

During runtime schema validation, PeopleSoft Integration Broker checks messages to ensure that the XML structure is valid according to the specified message schema. If Integration Broker encounters a service operation with a message structure that does not adhere to the specified message schema, message delivery fails and Integration Broker generates an error message. You can view the error details on the source application server, using the Asynchronous Details page or Synchronous Details page in the Service Operations Monitor.

When schema validation is enabled the structure of a message cannot be changed.

Prerequisites for Validating Message Schemas

Before you can enable schema validation, you must build or import a message schema for the message and message version.

If no message schema is present for service operations that contain regular nonrowset-based messages or container messages, it is not possible to enable validation.

For service operations that contain rowset-based messages, the system will attempt to generate a schema if one is missing. If the system is able to successfully generate a message schema for a rowset-based message, validation is enabled.

See Chapter 13, “Building Message Schemas,” page 265.

Copyright © 1988-2007, Oracle. All rights reserved.

Selecting Service Operations

This section discusses how to select service operations that contain messages to validate against message schemas at runtime.

Pages Used to Select Service Operations

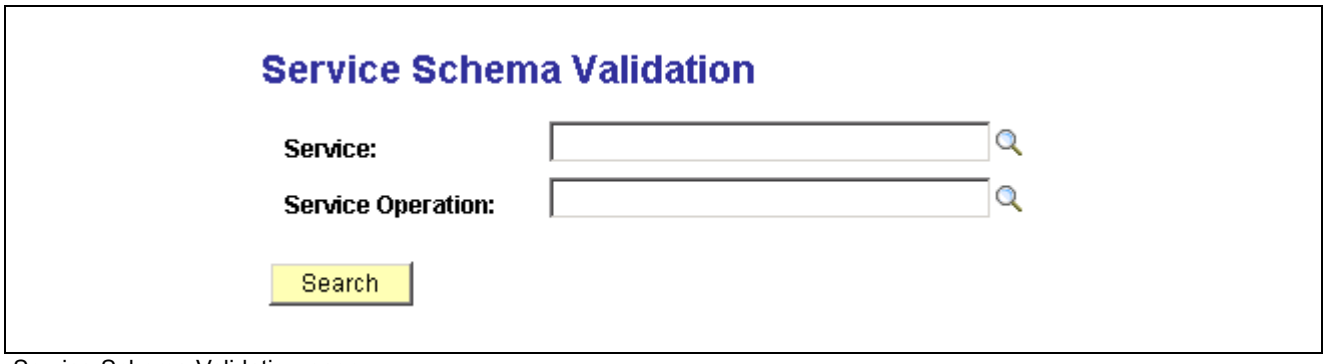
Page Name	Object Name	Navigation	Usage
Service Schema Validation page	IB_SERVICEVAL	Select PeopleTools, Integration Broker, Service Utilities, Service Schema Validation.	Use the page to: • Search for a service operation that contains messages to validate against message schemas at runtime. • Enable message schema validation.

Selecting a Service Operation

The first step to enabling runtime message schema validation is to select the service operations that contain the messages to validate.

When you access the Service Schema Builder component (IB_SERVICEVAL), the Service Schema Validation page appears and displays a search engine that you use to search for service operations.

To access the Service Schema Validation page select PeopleTools, Integration Broker, Service Utilities, Service Schema Validation.

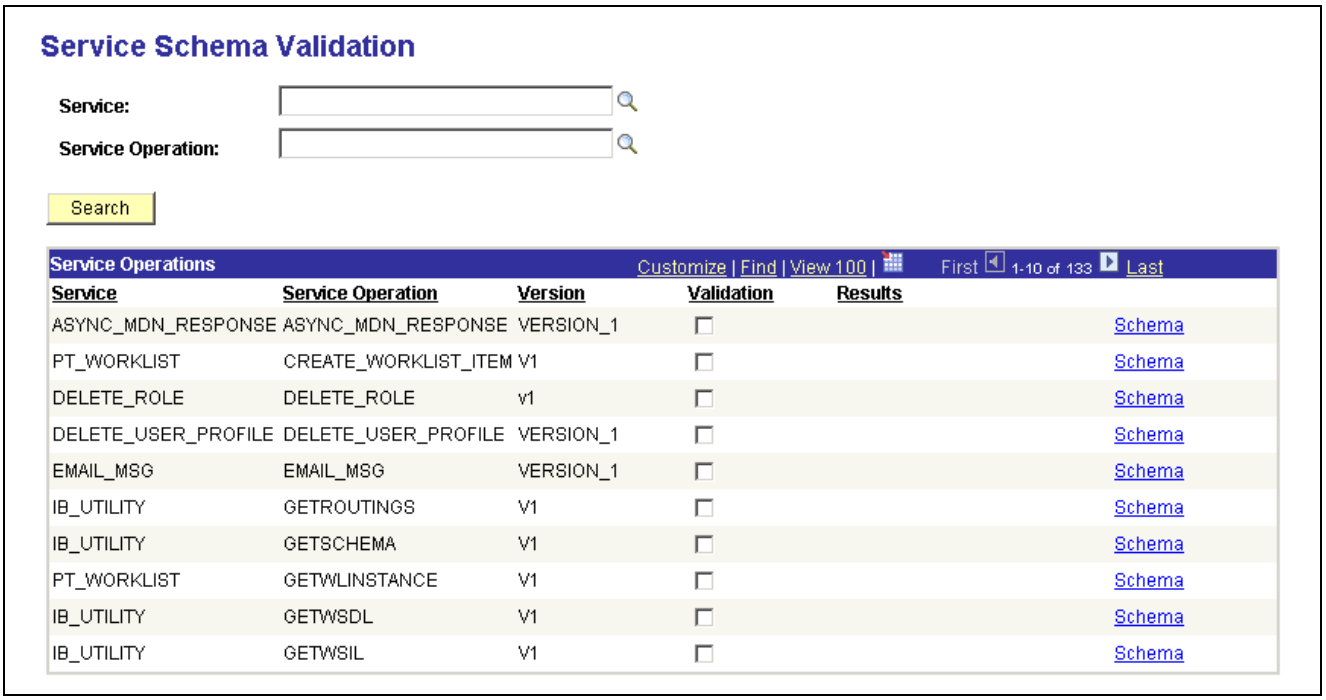


Service Schema Validation page

To search for a service operation, enter the service or service operation with which to work and click the Search button. A list of results displays in the Service Operations grid. If you do not enter any search criteria and click the Search button, the system returns all services and service operations in the database.

When you search for service schema validation data, the system returns the results in the Service Operations grid shown in the following example:

Copyright © 1988-2007, Oracle. All rights reserved.



Service Operations grid

Service Indicates the name of the service.

Service Operation Indicates the name of the service operation.

Version Indicates the version of the service operation.

Validation Indicates if runtime schema validation is enabled.

When the check box is checked, schema validation is enabled.

Results Displays validation results.

The valid values are:

- *Error generating schema. Unable to turn on validation.*

  This message appears if one or more of the messages in the service operation is nonrowset-based and schemas do not exist for the nonrowset-based messages.

- *Service operation saved.*

  This message appears when you have successfully enabled runtime schema validation and saved the changes.

- *Error saving service operation.*

Schema Click the link to access message schemas for all messages defined on the service operation.

See Chapter 16, "Enabling Runtime Message Schema Validation," Viewing Defined Message Schemas, page 318.

Copyright © 1988-2007, Oracle. All rights reserved.

Viewing Defined Message Schemas

This section discusses how to view defined XML schemas for messages.

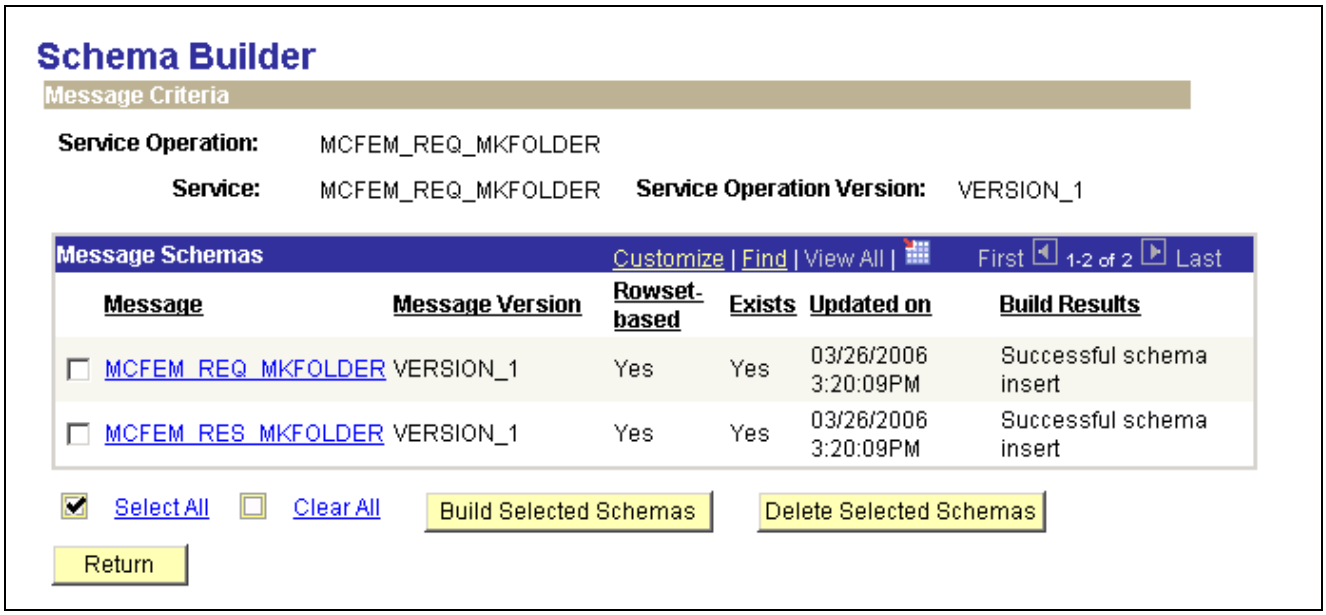
Pages Used to View Defined Message Schemas

Page Name	Object Name	Navigation	Usage
Service Schema Validation page	IB_SERVICEVAL	Select PeopleTools, Integration Broker, Service Utilities, Service Schema Validation.	Search for a service operation that contains messages that have generated XML schemas to view.
Schema Builder page	IB_SCHEMABUILD	Click the Schema link in the results grid of the Service Schema Validation page.	Select XML schemas to view.
Schema page	IB_SCHEMABUILD_SEC	Click a message name on the Schema Builder page.	View the XML schema for a message.

Viewing XML Schemas Defined for Messages

To view defined message schemas for all messages contained in a service operation, in the Service Operations grid, locate a service operation with which to work and click the Schema link.

The systems displays the service operation in the Schema Builder page. The following example shows the Schema Builder page displaying two messages for the *MCFEM_REQ_MKFOLDER* service operation.



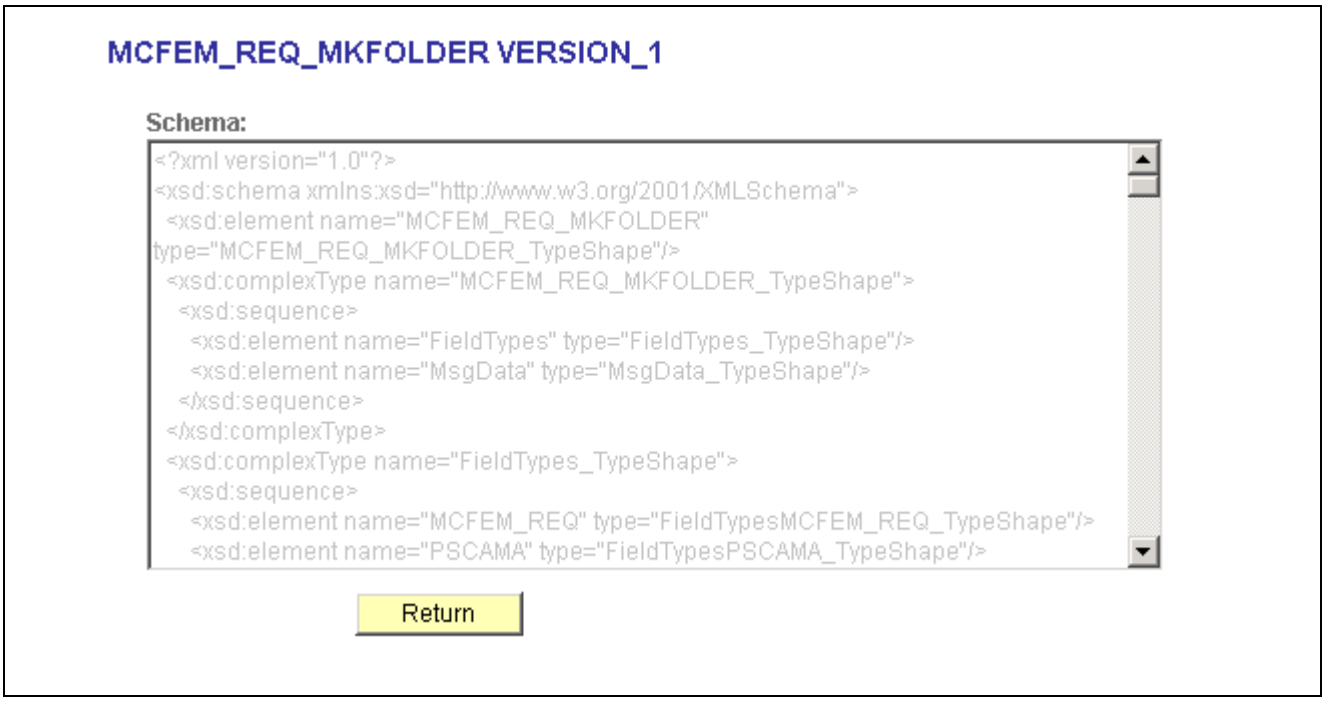
The Schema Builder page displaying messages for the MCFEM_REQ_MKFOLDER service operation

The Exists field displays a value of *Yes* for both messages and indicates schemas have been built for both messages.

If schemas are not built for a message or messages, you can build them directly from this page by selecting the check boxes next to each message name and clicking the Build Selected Schemas button.

Copyright © 1988-2007, Oracle. All rights reserved.

To view the XML schema for a message, click the message name link. The following example shows the XML schema for the *MCFEM_REQ_MKFOLDER* message.



XML schema for the MCFEM_REQ_MKFOLDER message.

To return to the Schema Builder page, click the Return button.

Using the Schema Builder page is documented in detail elsewhere in this PeopleBook.

See Also

Chapter 13, “Building Message Schemas,” page 265

Enabling Runtime Message Schema Validation

This section discusses how to enable runtime schema validation for all messages defined in a service operation:

Page Used to Enable Runtime Message Schema Validation

Page Name	Object Name	Navigation	Usage
Service Schema Validation page	IB_SERVICEVAL	Select PeopleTools, Integration Broker, Service Utilities, Service Schema Validation.	Enable runtime message schema validation for all messages defined in a service operation.
Service Operations page	IB_SERVICE	Select PeopleTools, Integration Broker, Integration Setup, Service Operations.	Enable runtime message schema validation for all messages defined in a service operation.

Copyright © 1988-2007, Oracle. All rights reserved.

Enabling Runtime Message Schema Validation

You can enable runtime message schema validation from the Service Schema Validation page or from the Service Operations page.

Using the Service Schema Validation Page to Enable Runtime Message Schema Validation

To enable runtime schema validation using the Service Schema Validation page:

1. Access the Service Schema Validation page.
2. Select a service operation that contains messages against which you want to validate message schemas.

   See Chapter 16, “Enabling Runtime Message Schema Validation,” Selecting Service Operations, page 316.
3. Check the Validation check box.
4. Click the Save button.

Using the Service Operations page to Enable Runtime Message Schema Validation

To enable runtime schema validation using the Service Operations page:

1. Access the Service Operations page.
2. In the Default Service Operation Version section of the page, check the Runtime Schema Validation check box.
3. Click the Save button.

 Copyright © 1988-2007, Oracle. All rights reserved.

CHAPTER 17

Creating Component Interface-Based Services

This chapter discusses how to:

- Select component interfaces to expose as services.
- Select component interface methods to include in service operations.
- Generate component interface-based services and service operations.
- View component interface-based service definitions.

Understanding Creating Component Interface-Based Services

PeopleSoft Integration Broker enables you to take an existing component interface and create a service that can be used to invoke the component interface. Further, it creates service operations, including request messages and response messages (if appropriate). The system creates an inbound any-to-local routing for the service operation version, as well as handlers for each method you choose to include in the service.

All service operations you generate from component interfaces are synchronous service operations.

Naming Conventions Integration Metadata Created

This section highlight the naming conventions that the PeopleSoft system uses when it creates services and services-related data based on component interfaces.

When it creates a web service from a component interface, the PeopleSoft system adds a *CI_* prefix to the component interface name. So if the component interface name is *MYCI*, the service name the system creates is *CI_MYCI*

The following table highlights naming conventions the PeopleSoft systems applies to other services data based on the method with which you are working:

Component Interface Method	Service Operation Name	Message Name	Component Interface Handler	Request Message Shape	Response Message Shape
Create	<service_name>_C	Mxxxxxx	ONREQUESTHDLR	CI Buffer	Key
Find	<service_name>_F	Mxxxxxx	ONREQUESTHDLR	Get Keys	CI Buffer
Get	<service_name>_G	Mxxxxxx	ONREQUESTHDLR	Find Keys	Key Collection
Update	<service_name>_UP	Mxxxxxx	ONREQUESTHDLR	CI Buffer	Notification

Copyright © 1988-2007, Oracle. All rights reserved.

Component Interface Method	Service Operation Name	Message Name	Component Interface Handler	Request Message Shape	Response Message Shape
Updatedata	<service_name>_UD	Mxxxxxx	ONREQUESTHDLR	CI Buffer	Notification
User-defined	<service_name>_<method_name>	Mxxxxxx	ONREQUESTHDLR	CI Buffer	Method Return Type

The naming convention used for message names, *Mxxxxxx*, is the letter M followed by a random six-digit number, as denoted by the x's. An example of a message name is *M548902*.

Note. The maximum number of characters for a service operation name is 30. If using a user-defined method name yields a greater result, the name is used is <service_name>_Mxxx, where xxx is a three-digit random number. An example of such a name is *CI_USERCI_M101023*.

User-Defined Method Restrictions

All user defined methods that will be accessed via the service interface must have a doc string.

The doc string contains:

1. The keyword *GET* or *CREATE*.
2. An ordered list of parameter names corresponding to the method's parameters.

The following example shows the format of the doc string:

```
Function simpleFunction(&sParm As string, &dParm As date,
                        &nParm As number, &bParm As boolean)
                        Returns string
   Doc "GET, StringParm, DateParm, NumParm , BoolParm"

   Local string &aString = &sParm;
   &aString = &aString | " -- " | &dParm;
   &aString = &aString | " -- " | &nParm;
   &aString = &aString | " -- " | &bParm;
   Return &aString;

End-Function;
```

The contents of the doc string are used when the function is invoked. If GET is specified, then the GET keys are set, and GET is called on the component interface before the used-defined method is invoked. If CREATE is specified, the CREATE keys will be set and CREATE will be called before the method is invoked.

The list of parameters are used at runtime to match the data in the input message with the method's parameters. This is an ordered list; if the parameter list in the doc string and the method parameters don't match, then the method may not work correctly. The names in the doc parameter list will be the names visible in any WSDL created for the service.

Method parameters and return values must be of a primitive type, such as a string, date or number. Object parameters or return values are not supported.

Copyright © 1988-2007, Oracle. All rights reserved.

Impact of Changing Component Interfaces

If a user modifies or deletes a component interface in PeopleSoft Application Designer, when the user saves the changes, PeopleSoft Integration Broker checks if a service exists for the component interface. If a service exists, and the component structure/properties have changed in the component interface, a warning message appears stating that a service exists for the component interface.

If the component interface structure changes, a status of *Does not match* appears in the CI-Based Services Review Status grid, and the operation appears as active in the Service Operations component.

Prerequisites

Prior to creating any component interface-based web services and service operations, you must define the schema namespace, service namespace, and target location in the Service Configuration page

See Chapter 14, "Managing Services," Configuring PeopleSoft Integration Broker for Handling Services, page 277.

Selecting Component Interfaces to Expose as Services

This section discusses how to select component interfaces to expose as web services.

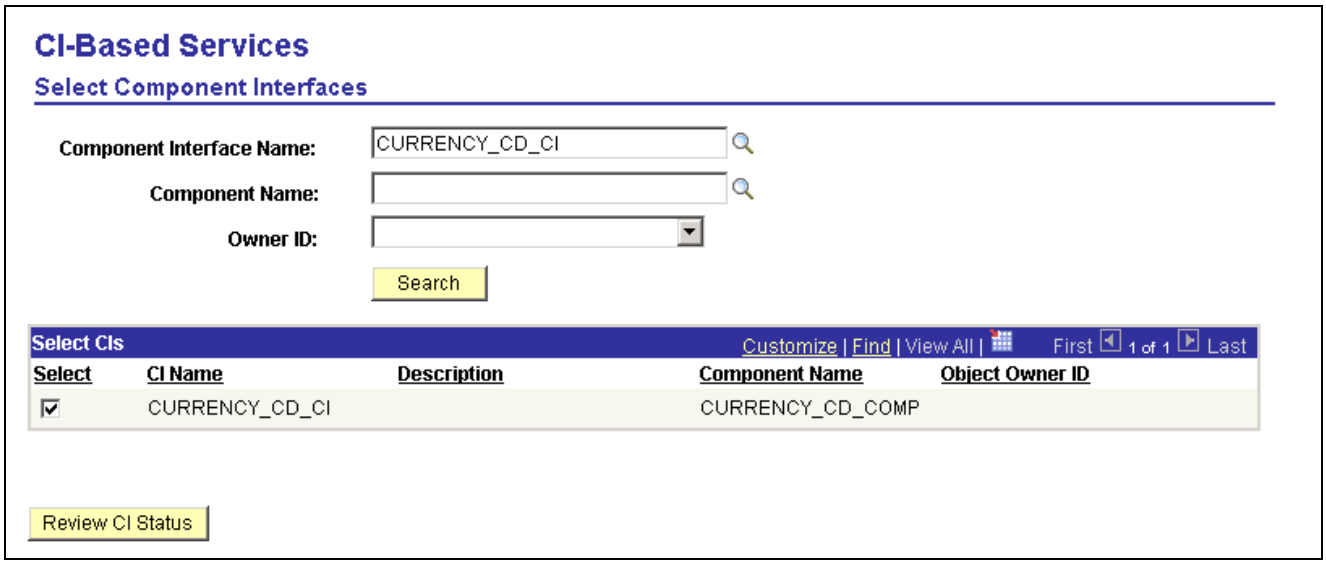
Page Used to Select Component Interfaces

Page Name	Object Name	Navigation	Usage
Select Component Interfaces	IB_CISERVICES	Select PeopleTools, Integration Broker, Web Services, CI-Based Services.	Choose component interfaces to expose as a web service.

Selecting Component Interfaces

The first step to creating component interface-based services is to choose the component interface on which to base a service. To do so, use the CI-Based Services component (IB_CISERVICES) and the CI–Based Services – Select Component Interface page. The following graphic shows this page:

Copyright © 1988-2007, Oracle. All rights reserved.



CI-Based Services–Select Component Interfaces page

When you search for component interfaces to select, the system returns results only for those component interfaces to which you have permissions.

To select a component interface:

1. Search for a component interface:

   Note. You can search only for those component interfaces to which you have permissions.

   - Click the Search button to search from all component interfaces in the database, or
   - Select one or more of the following criteria to narrow your search and then click the Search button.

   Component Interface Name Enter part or all of the name of the component interface to use, or click the Lookup button to search for one.

   Component Name Enter part or all of the component name to which a component interface belongs.

   Owner ID Owner ID dropdown list box, select the person or group that owns the component interface.

   The Select CIs grid displays are component interfaces that match your search.
2. Check the Select box next to one or more component interfaces.
3. Click the Review CI Status button.

The Review CI Status page appears where you can review details about the select component interface and select the methods to include as operations in the service.

Selecting Component Interface Methods to Include as Service Operations

Copyright © 1988-2007, Oracle. All rights reserved.

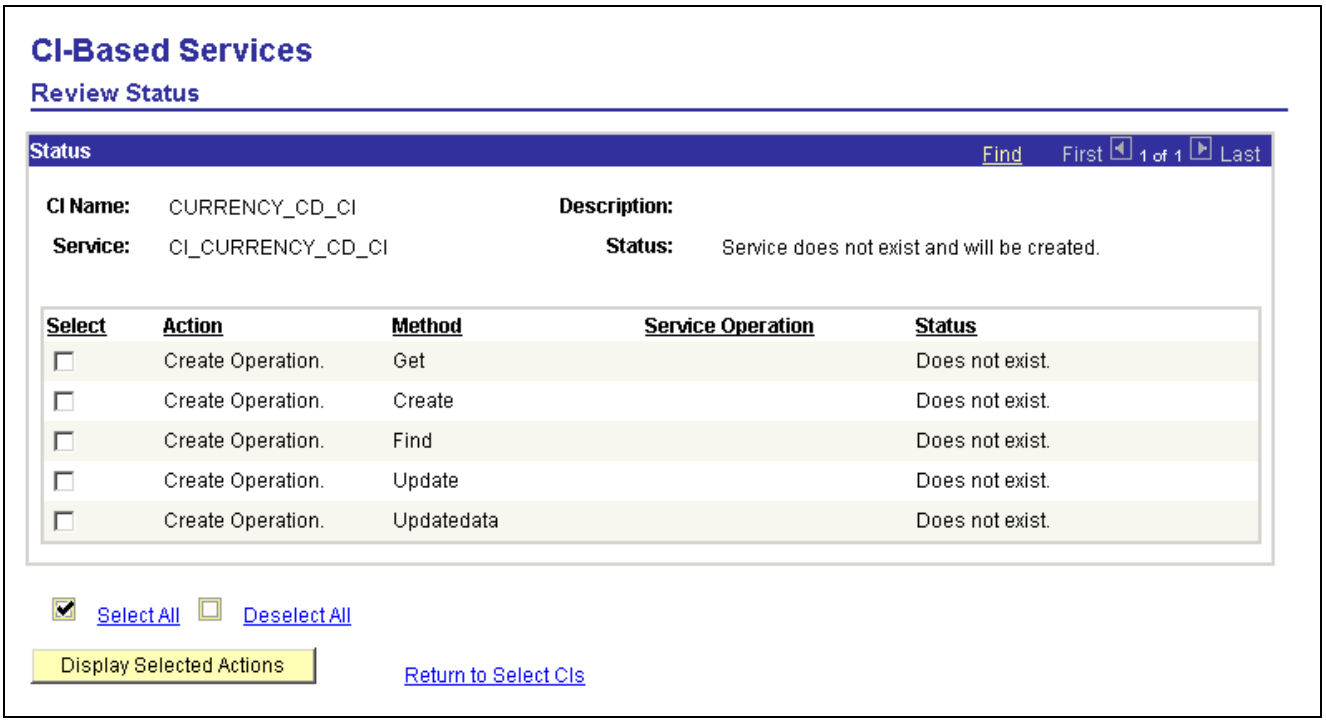
This section discusses selecting methods to include web services as service operations.

Page Used to Select Methods to Include as Service Operations

Page Name	Object Name	Navigation	Usage
Review Status	IB_CISERVICES2	From the CI-Based Services-Select Component Interfaces page, click the Review CI Status button.	Select component interface methods to include as service operations.

Selecting Methods to Include in Service Operations

The following example shows the CI-Based Services-Review Status page:



CI-Based Services–Review Status page

The Review Status page shows the following information about the component interface you select to expose as a service:

CI Name Name of the component interface.

Service Name of the service created based on the component interface.

Note that the service name is the component interface name with *CI_* added as a prefix.

Description A description of the component interface. This information is taken from the component interface record and displays if it exists.

Status(Service) Displays the status of the service. The valid values are:

- *Service does not exist and will be created.* Indicates that a services does not exist for the service and the system will create one.

Copyright © 1988-2007, Oracle. All rights reserved.

- *Service Exists*. Indicates that a service has already been created for the component interface.

Select Check the box to include a method as an operation for the service.

If the box is disabled, the method has already been included in the operation and the Service Operation field displays the name of the operation created.

You can check one or more methods at a time.

Action Displays the action available to perform on the method. The valid values are:

- *Create Operation*. This action displays if no service operation exists for the method.
- *Create New Version.*. This action displays if the current request or response shape do not match the shapes previously generated. A new service operation version is generated.
- *None*. If an operation exists, it compares the component interface and the service operation. If they are in sync, no action is required.

Method Name of the component interface method.

The system displays user-defined and standard methods. All user-defined methods appear in lowercase.

You can create service operations based on the following standard component interface methods:

- *Create*
- *Get*
- *Find*
- *Update*
- *Updatedata*
- User-defined methods.

Update and Updatedata appear if both Get and Save have been enabled in the component interface.

Note. All user-defined methods are lowercase. If you have a user-defined method called *update*, it is a different method than the Update method used here.

Service Operation Name of the service operation, if one exists for a method.

The name the system give the service operation depends on the service name as well as the component interface method.

Status(Service Operation) The status corresponds to the value in the Action field. The valid values are:

- *Does not exist*. No service operation exists for the method.
- *Does not match*. The service operation does not match the existing component interface.
- *OK*. The service operation matches the existing component interface.
- *Operation created.* The system created the operation.

 Copyright © 1988-2007, Oracle. All rights reserved.

Display Selected Actions Click the button to display a summary of the actions requested and then generate services and service operations.

Generating Component Interface-Based Services

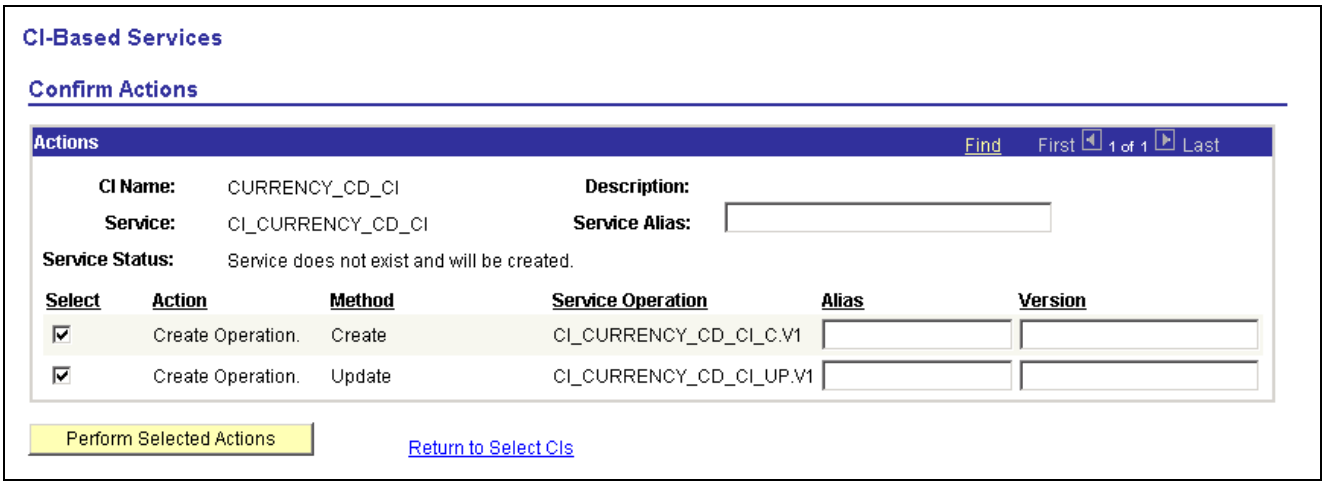
This section discusses generating service and service operations from component interfaces and component interface methods.

Page Used to Generate Component Interfaced-Based Services

Page Name	Object Name	Navigation	Usage
Confirm Actions	IB_CISERVICES_SEC	From the CI-Based Services-Review Status page, click the Perform Selected Actions button.	Confirm component interface methods to include in service operations for a service and generate the service and operations.

Generating Services and Service Operations

The following example shows the CI-Based Services–Confirm Actions page:



CI-Based Services–Confirm Actions page

You can work with the following page elements:

Service Alias (Optional.) Enter an alias name for the service. The name you enter can be lower or mixed-case. If specified, this is the service name that appears in any WSDL documents you generate.

Select (Optional.) Clear the box to omit creating a service operation for a method.

Alias(Service Operation) (Optional.) Enter an alias name for the service operation. The name you enter can be lower or mixed-case. If specified, this is the service operation name that appears in any WSDL documents you generate.

Version (Optional.) Service operations created default version *V1*.

Copyright © 1988-2007, Oracle. All rights reserved.

You may enter a different value to use as the version when the service operation is created.

This field is a text field, so you may enter numeric or text values.

Perform Selected Action Click the button to generate services and service operations for the component interface and selected methods.

After the service and service operations are created, the CI-Based Services–Review Status page displays and you can review the actions performed as well as access the service definition created.

Return to Select CIs Click the link to return to the previous page, the CI-Based Services–Review Status page.

Viewing Component Interface-Based Service Definitions

This section discusses viewing service definitions generated from component interfaces.

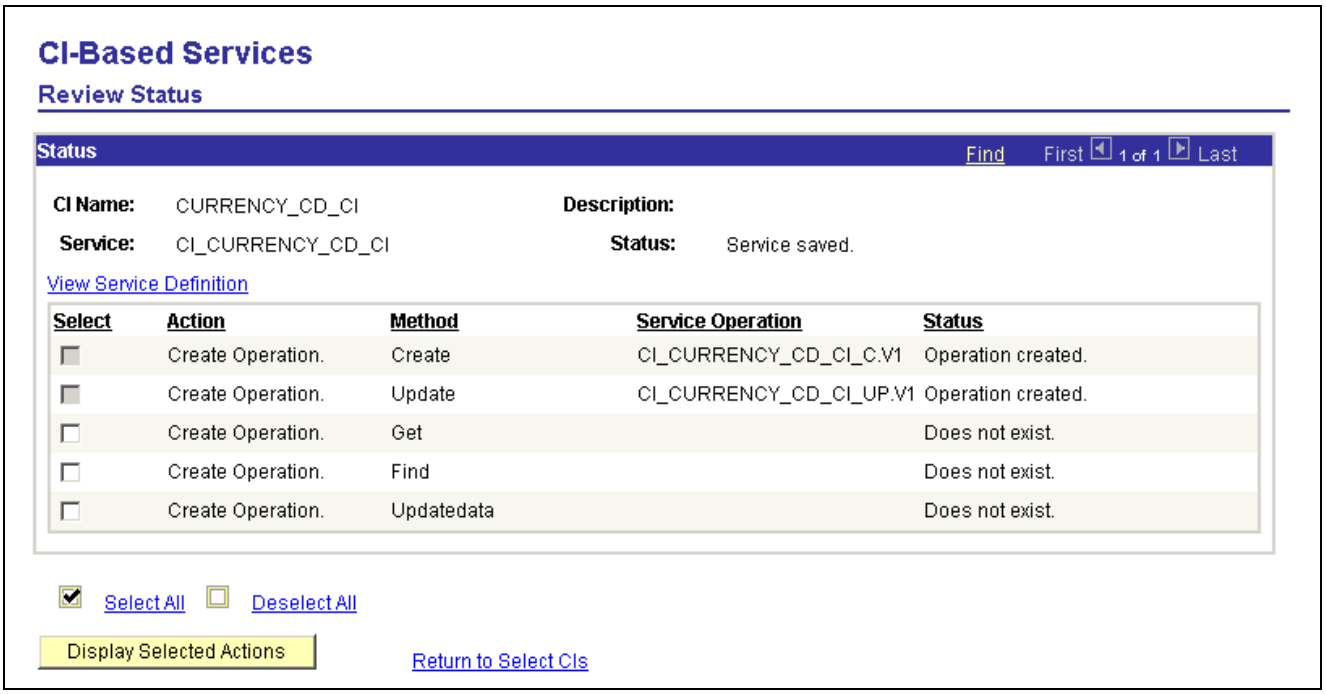
Pages Used to View Component Interface-Based Service Definitions

Page Name	Object Name	Navigation	Usage
Review Status page	IB_CISERVICES2	On the CI-Based Services-Review Status page, click the Perform Selected Actions button.	Use this page to review the actions performed.
Services page	IB_SERVICEDEFN	On the CI-Based Services-Review Status page, after you click the Perform Selected Actions button to generate the component interface-based service, click the View Service Definition link.	View the service definition for the component interface-based service.

Viewing Component Interface-Based Service Definitions

After you generate services and service operations, the following CI-Based Services–Review Status page appears:

 Copyright © 1988-2007, Oracle. All rights reserved.



CI-Based Services–Review Status page for generated service and service operations

Use this page to review the actions performed. For example, the previous graphic shows the names of the two service operations created as well as the service operation status of *Operation created*.

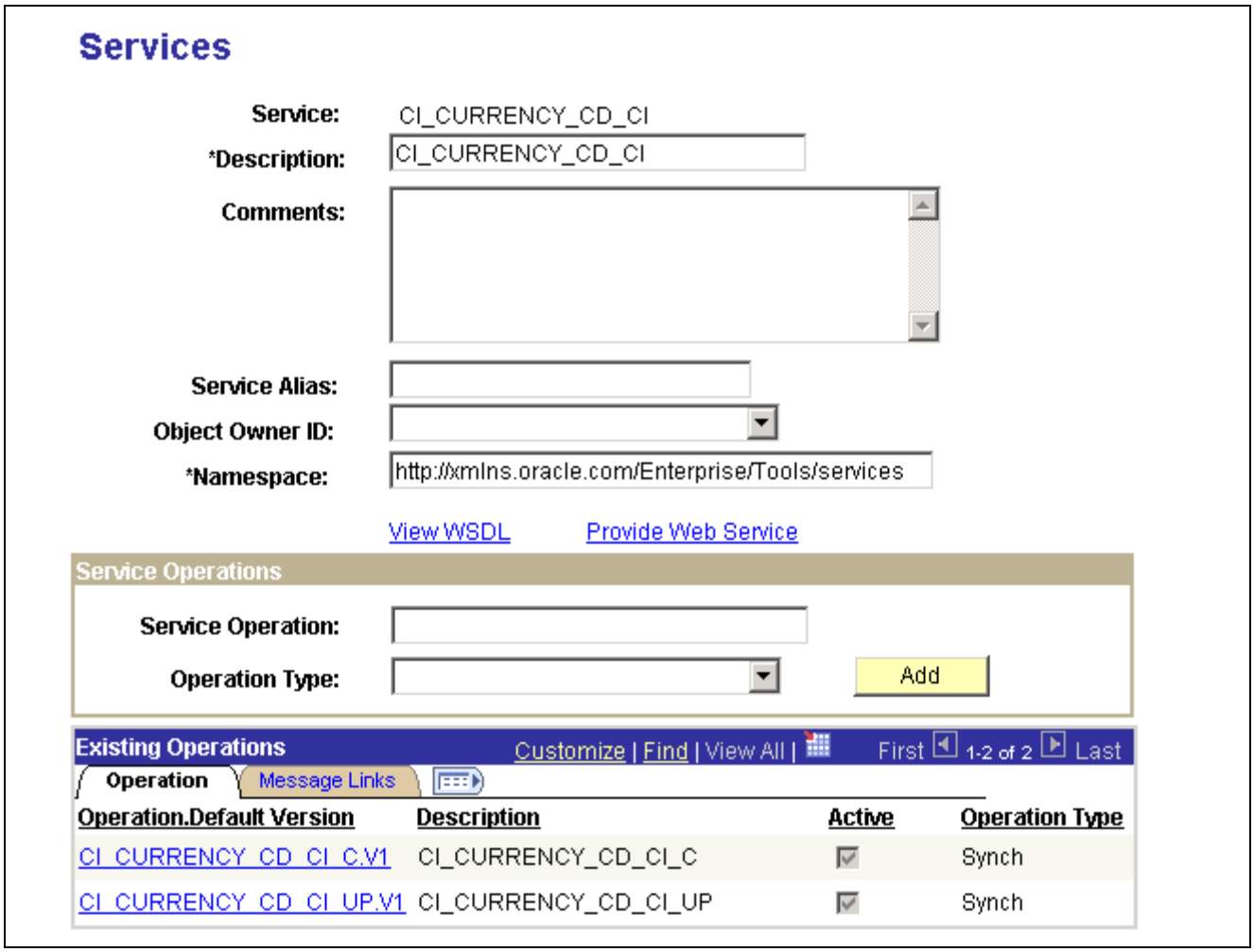
From this page you can continue to create additional service operations using the remaining available methods for the component interface.

Or, you can click the Return to Select CI link to return to the CI-Based Services–Select Component Interface page to select new component interfaces for which to generate services and service operations.

You can also click the View Service Definition link to view the service definition for the service created.

When you click the View Service Definition link the service you created, *CI_CURRENCY_CD_CI* appears in the Services page as shown in the following graphic:

Copyright © 1988-2007, Oracle. All rights reserved.



Services page for the CI_CURRENCY_CD_CI service

From this page you can perform actions as you would on any other service, including:

- Click the Provide Web Service link to generate WSDL for the service.
- Use the Service Operations box to add additional service operations to the service.
- Click either of the operations that display in the Existing Operations box to generate routing definitions, view the response, request or fault messages, view handler details, and more.
- And so on.

See Also

Chapter 14, “Managing Services,” page 275

Chapter 15, “Managing Service Operations,” page 291

 Copyright © 1988-2007, Oracle. All rights reserved.

CHAPTER 18

Managing Routing Definitions

This chapter discusses how to:

- View routing definitions.
- Manage system-generated routing definitions.
- Create routing definitions.
- Use introspection to create routing definitions.
- Activate and inactivate routing definitions.
- Rename and delete routing definitions.
- Delete duplicate routing definitions.

Understanding Routing Definitions

This section provides overview information about routing definitions

Routing Definitions

A routing definition defines the sending and receiving nodes for a transaction, specifies any inbound and outbound transformations to invoke and defines external aliases. It also defines overrides that the default integration gateway and the default target connector that the local node use to communicate with an integration endpoint.

Routing Types

There are three routing types:

Any-to-local An any-to-local routing enables the local node to receive transactions from any node.

This routing type is for inbound transactions only and the “any” node is always the sending node.

You can use this routing type for all service operation types, except for asynchronous-to-synchronous service operations.

Local-to-local A local-to-local routing is a routing in which transactions are sent and received within the local database.

Point-to-point A point-to-point routing requires that specific nodes that you define send and receive a transaction.

Copyright © 1988-2007, Oracle. All rights reserved.

Defining Routing Definitions

A routing definition must exist on each system that is participating in a particular transaction.

For example, let's say that System A is going to send a service operation to System B and a point-to-point routing needs to be created between the two systems. Further, the local node on System A is called *Node A*, and the local node on System B is called *Node B*.

System A and System B need to have the identical service operation defined on each of their systems.

In addition, System A needs to have an outbound routing definition defined on its system that specifies its local node, *Node A*, as the sending node. The routing definition must specify System B's local node, *Node B*, as the receiving node.

System B needs to have an inbound routing definition defined on its system that specifies its local node, *Node B*, as the receiving node. The routing definition on System B also needs to specify System A's local node, *Node A*, as the sending node.

The exception to this is when a receiving system generates an any-to-local routing for a service operation. If the receiving system has an any-to-local routing definition defined for a particular service operation, it will accept requests from any node without the need of a specific point-to-point routing definition.

So using the examples of System A being the sending system and System B being the receiving system, this is what happens when an any-to-local routing is defined. System A still needs to have a routing definition defined on its system where its local node, *Node A*, is defined as the sending node, and System's B local node, *Node B*, is defined as the receiving node. On System B, when the any-to-local routing was generated, the PeopleSoft system automatically populated System B's local node, *Node B*, as the receiving node and listed the value of *~Any~* as the sending node to designate that the system will accept the service operation specified on the routing from any node.

Methods for Generating and Defining Routing Definitions

This section discusses the three methods of generating and defining routing definitions.

System-Generated Routing Definitions During Upgrade

If upgrading from a PeopleTools 8.47 or earlier release, during the upgrade process the PeopleSoft system creates routing definitions from node transaction and relationship data defined in your earlier PeopleTools 8.4x release.

System-Generated Routing Definitions During Consuming Services

When you consume a service using the Consume Web Service wizard, the system created PeopleSoft integration metadata for the imported service, including routing definitions.

System-Generated Routing Definitions at Runtime

The PeopleSoft system can generate any-to-local and local-to-local routing definition for you. The system takes integration metadata from the service operation, including service operation name, service operation version, service operation type, local node information, and other data, and generates a routing definition.

If you make any subsequent modifications to the service operation, you can regenerate the routing definition to reflect the changes. In addition, at any point you can open the definition and manually modify it to include transformations, as well as override default integration gateway and target connector settings.

You use the Service Operations-General page to generate any-to-local and local-to-local routing definitions.

 Copyright © 1988-2007, Oracle. All rights reserved.

You may also manually create local-to-local routing definitions. However, you must always system-generate any-to-local routing definition.

See Chapter 18, "Managing Routing Definitions," Managing System-Generated Routing Definitions, page 335.

User-Defined Routing Definitions

When you require a point-to-point routing, you may manually create the routing definition or generate it using introspection.

You can also manually create local-to-local routing definitions. However, you must always system generate any-to-local routing definitions.

To create user-defined routing definitions, use the Routings component or the Service Operations-Routings page.

See Chapter 18, "Managing Routing Definitions," Creating Routing Definitions, page 338.

Introspecting Nodes To Generate Routing Definitions

PeopleSoft provides the ability to introspect other PeopleTools 8.49 nodes (PeopleSoft and external nodes) to generate point-to-point routing definitions.

You use the Deployment and Validation component to introspect nodes.

See Chapter 18, "Managing Routing Definitions," Using Introspection to Create Routing Definitions, page 347.

Summary of Methods for Creating and Generating Routing Definitions

The following table summarizes the method for generating and defining routing definitions:

Routing Type	System-Generated	User-Defined	Introspection
Any-to-local	Yes	No	No
Local-to-local	Yes	Yes	No
Point-to-point	No	Yes	Yes

Routing Definition Naming Conventions

The following table lists the naming conventions for routing definitions:

Method for Generating and Creating Routing Definitions	Convention	Description
System-generated during upgrade.	~GEN_UPG~<unique number> For example: *~GEN_UPG~10062*	Routing definitions generated during the upgrade process. These may be any-to-local, local-to-local, or point-to-point routing definitions.

Copyright © 1988-2007, Oracle. All rights reserved.

Method for Generating and Creating Routing Definitions	Convention	Description
Routing definitions generated or created by: • System-generated at runtime • Node introspection.	~GENERATED~<unique number> For example: *~GENERATED~15312*	Routing definitions generated by the PeopleSoft system from the Service Operations-General tab or from the Deployment Validation component. When generated from the Service Operations-General tab, these routing definitions are any-to-local or local-to-local. When generated from the Deployment Validation component, routing definitions are usually point-to-point definition, but may also be local-to-local routing definitions.
System generated by the Consume Web Service wizard.	~IMPORTED~<unique number> For example:*~IMPORTED~14857*	Routing definitions generated using the Consume Web Service wizard.
User-defined.	Up to 30 characters, no spaces. For example: *QE_ROUTING*. No special characters, such as dots (“.”) and slashes (“/” or “\”), are permitted.	Manually created point-to-point routing definitions and local-to-local routings. You can also rename system-generated routing definitions or introspected routing definitions using the Service Administration component.

Routing Definition External Aliases

When working with routing definitions you have the option of creating a routing alias. This alias is used as a SOAPAction attribute in the WSDL binding to identify the service operation in the Integration Broker metadata.

The routing external alias defaults to <*ServiceOperationAlias*>.<*Version*>, if present. Otherwise it defaults to <*ServiceOperation*>.<*Version*>.

In an asynchronous request/response any-to-local routing, the outbound routing alias format is <*Alias Name*>*_CALLBACK.*<*Version*>.

For inbound transactions you can fire multiple service operations for one invocation when external aliases on the routing definition are the same for each service operation. This is called service operation mapping.

See Also

Chapter 18, “Managing Routing Definitions,” Service Operation Mapping, page 334

Service Operation Mapping

You can map inbound asynchronous transactions to one or more service operations by specifying the name of the inbound transaction as the external alias on the routing for each service operation that you want to invoke.

Note. Service operation mapping is supported for inbound asynchronous transactions only.

 Copyright © 1988-2007, Oracle. All rights reserved.

For example, there is an inbound asynchronous transaction from SAP called *Customer_SAP*. However, the service operation contained in that transaction maps to two service operations on the PeopleSoft system, *Customer_Get* and *Customer_Update*. To invoke both service operations, change the external alias name on the inbound routing definition for the *Customer_Get* and *Customer_Update* service operations to *Customer_SAP*. When the routings are determined at runtime for this external service operation name, PeopleSoft Integration Broker will find both service operations (*Customer_Get* and *Customer_Update*) and process them accordingly.

Viewing Routing Definitions

To view routing definitions defined for a service operation, use the Service Operations-Routings page.

See Chapter 15, "Managing Service Operations," Accessing and Viewing Service Operation Definitions, page 292.

Managing System-Generated Routing Definitions

This section discusses how to:

- View system-generated routing definition status.
- Initiate system-generated routing definitions.
- Regenerate system-generated routing definitions.

Understanding Managing System-Generated Routing Definitions

The PeopleSoft system can automatically generate any-to-local and local-to-local routing definitions when you save a service operation definition.

After the system generates the routing definition, you can view and fine-tune the definition as needed using the pages in the Routings component.

In addition, if you make any changes to a service operation after the system has generated a routing definition for it, you can have the system regenerate a routing definition. However, any modifications made to the routing definition are lost when you regenerate it.

Page Used to Manage System-Generated Routing Definitions

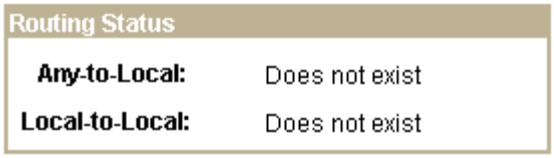
Page Name	Object Name	Navigation	Usage
Service Operations-General page	IB_SERVICE	Select PeopleTools, Integration Broker, Integration Setup, Service Operations.	Use this page to verify if any system-generated routing definitions exist for the service operation. Use this page to initiate and regenerate system-generated any-to-local and local-to-local routing definitions.

Copyright © 1988-2007, Oracle. All rights reserved.

Viewing System-Generated Routing Definition Status

The Service Operations-General page features a Routing Status box that you can use to verify if any system-generated routing definitions exist for a service operation. To access the page select PeopleTools, Integration Broker, Integration Setup, Service Operations.

The following example shows the Routing Status box:



Service Operations-General page Routing Status box

When an any-to-local or local-to-local routing definition exists for the service operation, the corresponding field displays a status of *Exists*. When no routing definition exists, the corresponding field displays *Does not exist*.

Initiating System-Generated Routing Definitions

The Default Service Operation Version section of the Service Operations-General page features a Routing Actions Upon Save box where you can choose the type of routing to generate, any-to-local or local-to-local. To access the page select PeopleTools, Integration Broker, Integration Setup, Service Operations.

The following example shows the Routing Actions Upon Save box:

Service Operations-General page Routing Actions Upon Save box

When you initiate system-generated any-to-local or local-to-local routings, PeopleSoft Integration Broker checks to see if the routing you are initiating is already in the system. This situation can arise when any-to-local and local-to-local routings are created in another database and are imported into the current database. If the routing already exists in the current database, a message appears indicating so and no new routing is generated. You must remove the routing before generating a new one.

See Chapter 18, "Managing Routing Definitions," Deleting Duplicate Routing Definitions, page 358.

To initiate a system-generated routing definition:

1. From the Service Operations-General page, locate the Default Service Operation Version section.
2. In the Routing Actions Upon Save group box select one of the following options:
   - *Generate Any-to-Local.* Generates an any-to-local routing definition when you save the service operation record.
   - *Generate Local-to-Local.* Generates a local-to-local routing definition when you save the service operation.
3. Click the Save button.

 Copyright © 1988-2007, Oracle. All rights reserved.

When you save the service operation the system generates the routing definition that you selected.

After you save the service operation definition the Routing Status group box displays a status of *Exists* for the routing definition generated.

To view the routing definition , click the Service Operations-Routings tab and click the name of the routing. The Routing Definitions page appears and you can view and modify routing definition details.

See Also

Chapter 18, “Managing Routing Definitions,” Routing Definition Naming Conventions, page 333

Chapter 18, “Managing Routing Definitions,” Defining General Routing Information, page 342

Chapter 18, “Managing Routing Definitions,” Viewing Routing Parameters for Requests and Responses, page 344

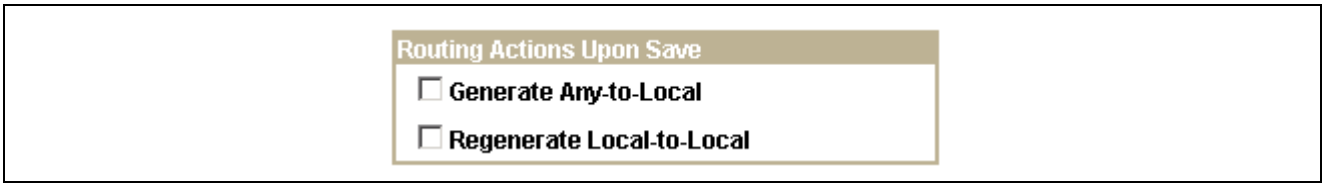
Chapter 18, “Managing Routing Definitions,” Overriding Gateway and Connector Properties, page 346

Regenerating System-Generated Routing Definitions

If a system-generated routing exists for a service operation and you change some aspect of the service operation, you can have the system regenerate the routing definition. However, any modifications made to the routing definition are lost when you regenerate it.

To initiate the regeneration of a routing definition, use the Routing Actions Upon Save box on the Service Operations-General page to regenerate the routing. To access this page select PeopleTools, Integration Broker, Integration Setup, Service Operations.

The following example shows the Routing Actions Upon Save box when a routing definition can be regenerated:



Regenerating a routing definition

To regenerate a system-generated routing definition:

1. From the Service Operations-General page, locate the Default Service Operation Version section.
2. In the Routing Actions Upon Save group box select one of the following options:
   - *Regenerate Any-to-Local.* Regenerates an any-to-local routing definition when you save the service operation definition.
   - *Regenerate Local-to-Local.* Regenerates a local-to-local routing definition when you save the service operation.
3. Click the Save button.

When you save the service operation the system regenerates the routing definition that you selected.

After you save the service operation record the Routing Status group box displays a status of *Exists* for the routing definition generated.

Copyright © 1988-2007, Oracle. All rights reserved.

Creating Routing Definitions

This section discusses how to:

- Add a routing definition.
- Define general routing definition information.
- View routing parameters for requests and responses.
- Override gateway and connector properties.

Understanding Creating Routing Definitions

This section provides an overview of routing definitions.

Routing Parameters for Requests and Responses

A routing definition contains routing parameters for each inbound request, inbound response, outbound request and outbound response associated with a service operation. The routing parameters that you can define include for each request or response include, routing alias, message names before and after transformation, and transformation program names.

You define routing parameters using the Routings-Parameters page in the Routings component.

The following tables list the number of routing parameters a routing definition contains based on the service operation type and whether the sending and receiving nodes are local.

Asynchronous service operations have the following routing parameters:

Service Operation Type	Sender is Local	Receiver is Local	Inbound Request Routing	Outbound Request Routing	Inbound Response Routing	Outbound Response Routing
Asynchronous One- Way	Y	Y	N	Y	N	N
Asynchronous One- Way	N	N	N	Y	N	N
Asynchronous One- Way	N	Y	Y	N	N	N
Asynchronous One- Way	N	N	Y	Y	N	N

Synchronous service operations have the following routing parameters:

 Copyright © 1988-2007, Oracle. All rights reserved.

Service Operation Type	Sender is Local	Receiver is Local	Inbound Request Routing	Outbound Request Routing	Inbound Response Routing	Outbound Response Routing
Synchronous	Y	Y	Y	Y	Y	Y
Synchronous	Y	N	N	Y	Y	N
Synchronous	N	Y	Y	N	N	Y
Synchronous	N	N	Y	Y	Y	Y

Asynchronous-to-synchronous service operations may have the following routing parameters:

Service Operation Type	Sender is Local	Receiver is Local	Inbound Request Routing	Outbound Request Routing	Inbound Response Routing	Outbound Response Routing
Asynchronous-to-synchronous	Y	Y	Y	N	N	Y
Asynchronous-to-synchronous	Y	N	N	Y	Y	N
Asynchronous-to-synchronous	N	Y	Y	N	N	Y
Asynchronous-to-synchronous	N	N	Y	Y	Y	Y

Asynchronous request/response service operations may have the following routing parameters:

Service Operation Type	Sender is Local	Receiver is Local	Inbound Request Routing	Outbound Request Routing	Inbound Response Routing	Outbound Response Routing
Asynchronous Request /Response	Y	Y	N	Y	Y	N
Asynchronous Request /Response	Y	N	N	Y	Y	N
Asynchronous Request /Response	N	Y	Y	N	N	Y
Asynchronous Request /Response	N	N	Y	Y	Y	Y

Copyright © 1988-2007, Oracle. All rights reserved.

Pages Used to Create Routing Definitions

Page Name	Object Name	Navigation	Usage
Routing Definitions	IB_ROUTINGDEFN	You can access this page using the following methods: • Select PeopleTools, Integration Broker, Integration Setup, Routings. • Select PeopleTools, Integration Broker, Integration Setup, Service Operations. Click the Routings tab. In the Routing Name field, add a name for a new routing definition and click the Add button.	Use the Routing Definitions page to add a routing record to the system. Also use this page to define or modify general information about the routing, including the service operation (and version) for which the routing is defined, the sending node and the receiving node. Use this tab to also activate and inactivate routing definitions.
Parameters page	IB_ROUTINGDEFNDOC	Select PeopleTools, Integration Broker, Integration Setup, Routings. Click the Parameters tab.	Use the Parameters page to view routing parameters for individual transaction requests and responses associated with the service operation. Information you can specify includes external aliases for requests and responses. Use this page to also specify any transformations the system is to invoke on the inbound or outbound side of a transaction, as well as the message name and version after transformation.
Connector Properties	IB_ROUTINGDEFNCON	Select PeopleTools, Integration Broker, Integration Setup, Routings. Click the Connector Properties tab.	Use the Connectors page to override the default integration gateway and target connector that the local node uses for communicating with an integration endpoint. **Note.** The Connector Properties page displays in the Routings component only if the receiving node is not the local node.

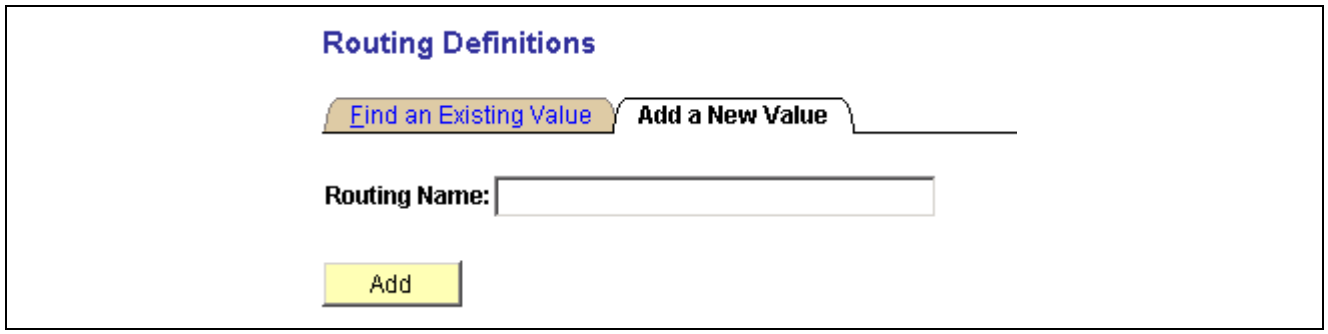
Adding Routing Definitions

You can add routing definitions to the PeopleSoft system from the Routings Definition page in the Routings component (IB_ROUTINGDEFN). You can also add them from within a service operation record from the Service Operations-Routings tab.

 Copyright © 1988-2007, Oracle. All rights reserved.

Adding Routing Records Using the Routings Component

The following graphic shows the page used to add a routing record when using the Routings component:



Adding a routing record from the Routings component

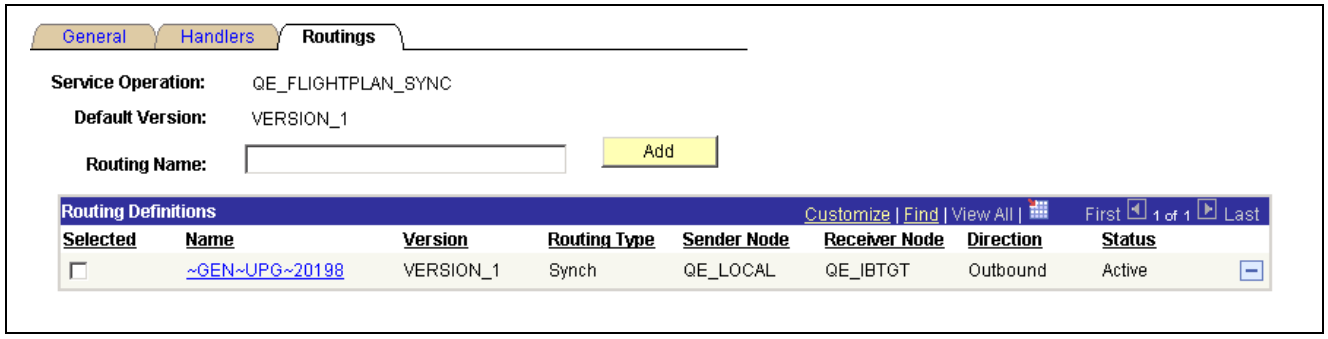
To add a routing record using the Routings component:

1. Select PeopleTools, Integration Broker, Integration Setup, Routings.
2. Click the Add a Value tab.
3. In the Routing Name field, enter a name for the routing definition.
4. Click the Add button.

The routing definition is added to the system and the Routing Definitions page appears where you can define the routing details.

Adding Routing Definitions From Service Operation Definitions

The following graphic shows the page used when adding a routing definition from the Service Operations-Routing tab of a service operation definition.



Adding a routing record from the Service Operations-Routings tab

To add a routing definition from a service operation definition:

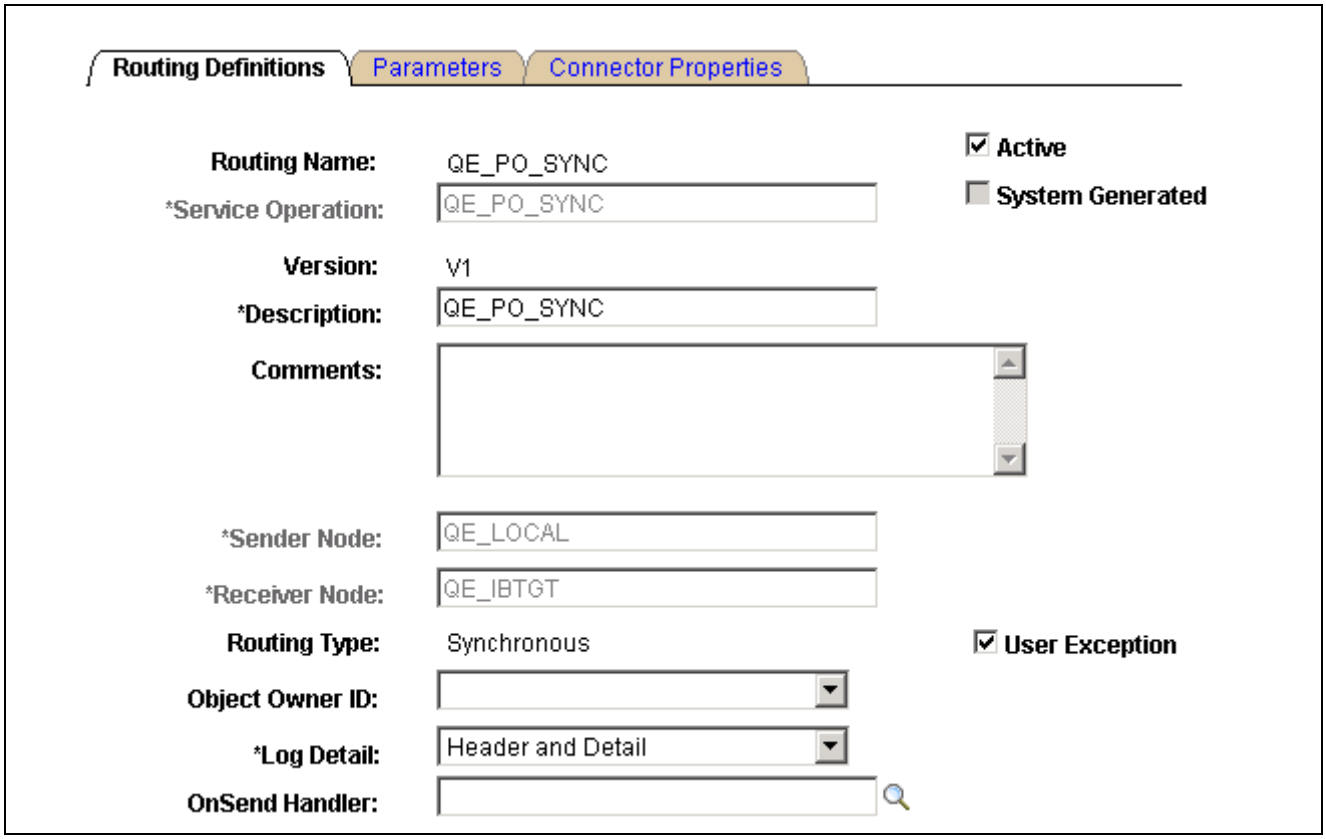
1. From within a service operation definition, click the Routings tab.
2. In the Routing Name field, enter a name for the routing definition.
3. Click the Add button.

The routing definition is added to the system and the Routing Definitions page appears where you can define the routing details.

Copyright © 1988-2007, Oracle. All rights reserved.

Defining General Routing Information

After you add a routing definition to the system use the pages of the Routing component to define the routing details. The following graphic show the Routing Definitions page used to define general routing information, as accessed from the Service Operations-Routings page.



Routing Definitions page

When you add a routing definition from a service operation record, the PeopleSoft system automatically populates some of the data on this page based on the data in the service operations record from which you created the routing. Automatically populated data includes the service operation name, version, and routing type.

Routing Name Indicates the name of the routing definition. This name is specified when you add a routing definition to the system.

Service Operation Enter the name of the service operation that will use the routing.

If you access the Routings component from the Service Operations-Routing tab, PeopleSoft Integration Broker automatically populates this information.

Active (Optional.) Check the box to activate the routing.

By default, new routing definition are active.

If any of the referenced nodes are inactive, you cannot activate the routing.

System Generated When selected, indicates that the PeopleSoft system generated the routing definition.

Version Indicates the version of the service operation selected.

Copyright © 1988-2007, Oracle. All rights reserved.

Description Description of the routing definition.

Comments (Optional.) Enter comments about the routing definition.

Sender Node Enter the name of the sending node.

Receiving Node Enter the name of the receiving node.

Routing Type Indicates the service operation type. PeopleSoft Integration Broker automatically populates this information when you select the service operation.

User Exception The User Exception check box displays only for synchronous service operations.

Check the box to enable exception handling using PeopleCode.

When enabled and an error occurs you can handle any errors in the calling PeopleCode.

If not enabled any errors that occur cause the program to stop.

Object Owner ID (Optional.) From the dropdown list box, select the owner of the definition.

The owner ID helps to determine the application team that last made a change to the definition. The values in the dropdown list box are translate table values that you can define in the OBJECTOWNERID field record.

Log Detail The Log Detail dropdown list box displays only for synchronous service operations.

This option enables you to set the level of information logging for synchronous messages that is viewable in the Service Operations Monitor.

The valid values are:

- *Header*.

  Log header information only. With this option, you can view synchronous message header information in the Service Operations Monitor.

- *Header and Detail*. Log header and message detail information. With this option, you can view synchronous message header information and XML message content on in the Service Operations Monitor.

  (Default.)

- *No Logging*. (Default.) Turn off all logging. No information is available to view in the Service Operations Monitor.

OnSend Handler This field displays when an OnSend handler is defined for the service operation and the sending node is the local node. It also displays when you the system is serving as a hub, and neither the sender nor receiver are local.

Select a handler from the list. This handler runs when a message is sent or received to perform processing logic.

OnReceive Handler This field displays when an OnReceive handler is defined for the service operation and:

- The sending node is the local node.
- The service operation type is asynchronous request/response where the sender is not local and the receiver is local.

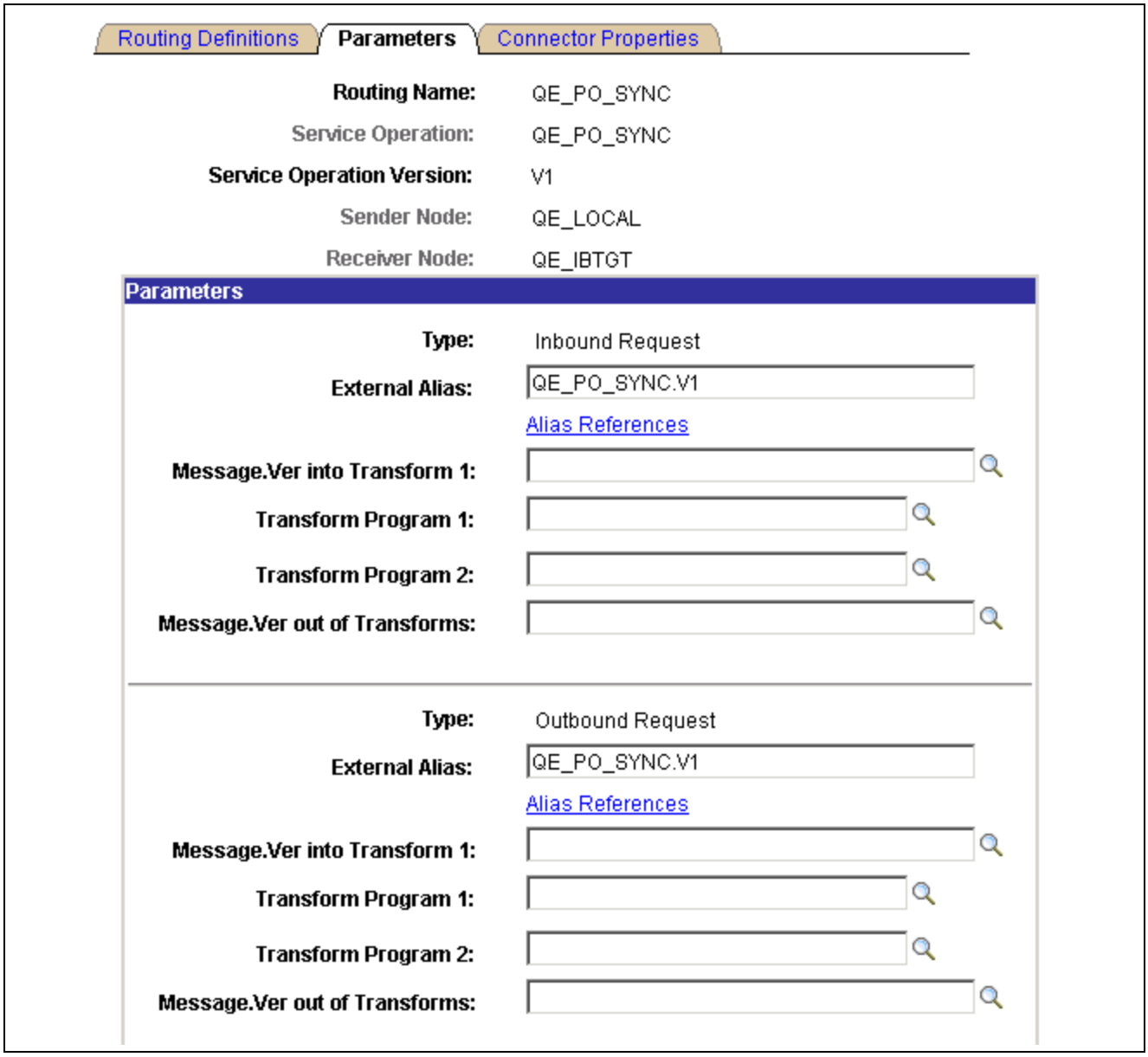
Copyright © 1988-2007, Oracle. All rights reserved.

- The system is serving as a hub, and neither the sender nor receiver are local.

Select a handler from the list. This handler runs when a message is sent or received to perform processing logic.

Viewing Routing Parameters for Requests and Responses

Use the Routings-Parameters page to view parameters for requests and responses associated with a service operation. Information you define includes, routing external aliases for inbound and outbound requests and responses, as well as any inbound or outbound transformations to invoke. The following graphic shows the Routings-Parameter page:



Routings-Parameters page

The following page elements display on the Routings-Parameters page:

Copyright © 1988-2007, Oracle. All rights reserved.

Type Specifies the routing direction and the type of message (request or response) associated with the service operation. This information is automatically populated from the service operation definition.

External Alias This alias is used as a SOAPAction attribute in the WSDL binding to identify the service operation in the Integration Broker metadata.

The routing external alias defaults to <*ServiceOperationAlias*>.<*Version*>, if present. Otherwise it defaults to <*ServiceOperation*>.<*Version*>.

In an asynchronous request/response any-to-local routing, the outbound routing alias format is <*Alias Name*>_*CALLBACK.*<*Version*>.

For inbound transactions you can fire multiple service operations for one invocation when external aliases on the routing definition are the same for each service operation. This is called service operation mapping.

Duplicate external aliases are not allowed for synchronous operations.

See Chapter 18, "Managing Routing Definitions," Service Operation Mapping, page 334.

Alias References Click the link to view other routing definitions with the same external alias.

Message.Ver info Transform 1 Displays the name of the request or response message associated with the service operation before any transformations are applied.

For inbound transactions, this is the message name and version as it arrives from the integration partner system, before any transformations are applied.

For outbound transactions, this is the message name and version directly from the PeopleSoft system, before any transformations are applied.

Transform Program 1 (Optional.) Enter the name of the transform program to invoke on the message listed in the Message.Ver info Transform 1 field.

Transform Program 2 (Optional.) Enter the name of the transform program to invoke after the transform program in the Transform Program 1 field has completed processing.

Note. When you invoke two transform programs, the output from the first transform program (Transform Program 1) is used as the input into the second transform program (Transform Program 2).

Message.Ver out of Transforms (Optional.) Enter the name of the message after all transform program have completed processing.

For inbound messages, this is the message name and version that the PeopleSoft system is expecting.

For outbound messages, this is the message name and version that the integration partner system is expecting.

Note. When the Routings-Parameters page first displays values for the Message.Ver info Transform 1 and Message.Ver out of Transforms fields display values to assist you in choosing transform programs. After you save the page, values do not appear in these fields unless the transform programs have an input/output messages associated with them.

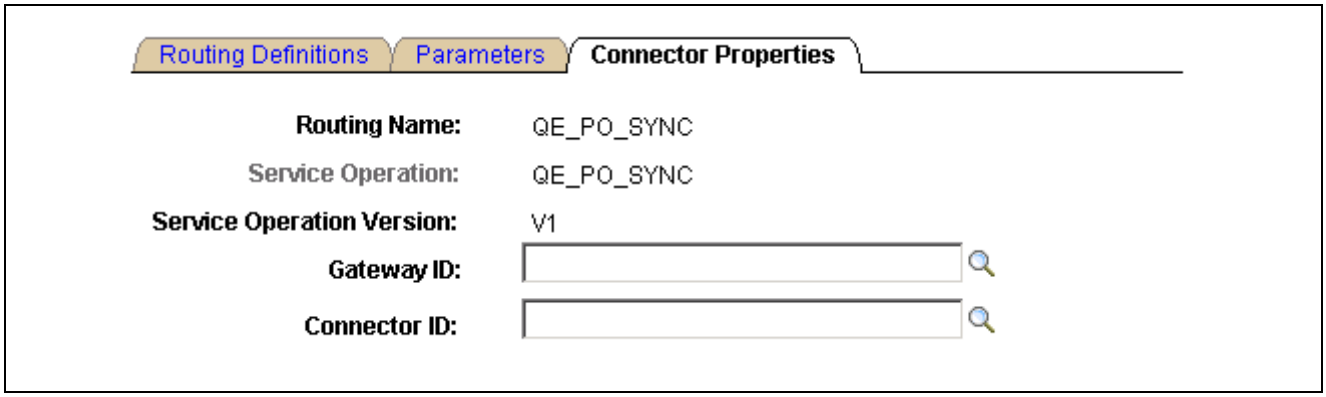
Copyright © 1988-2007, Oracle. All rights reserved.

Overriding Gateway and Connector Properties

The Routings-Connector Properties page enables you to override the default integration gateway and target connector that the local node uses to communicate with an endpoint for a specific routing definition.

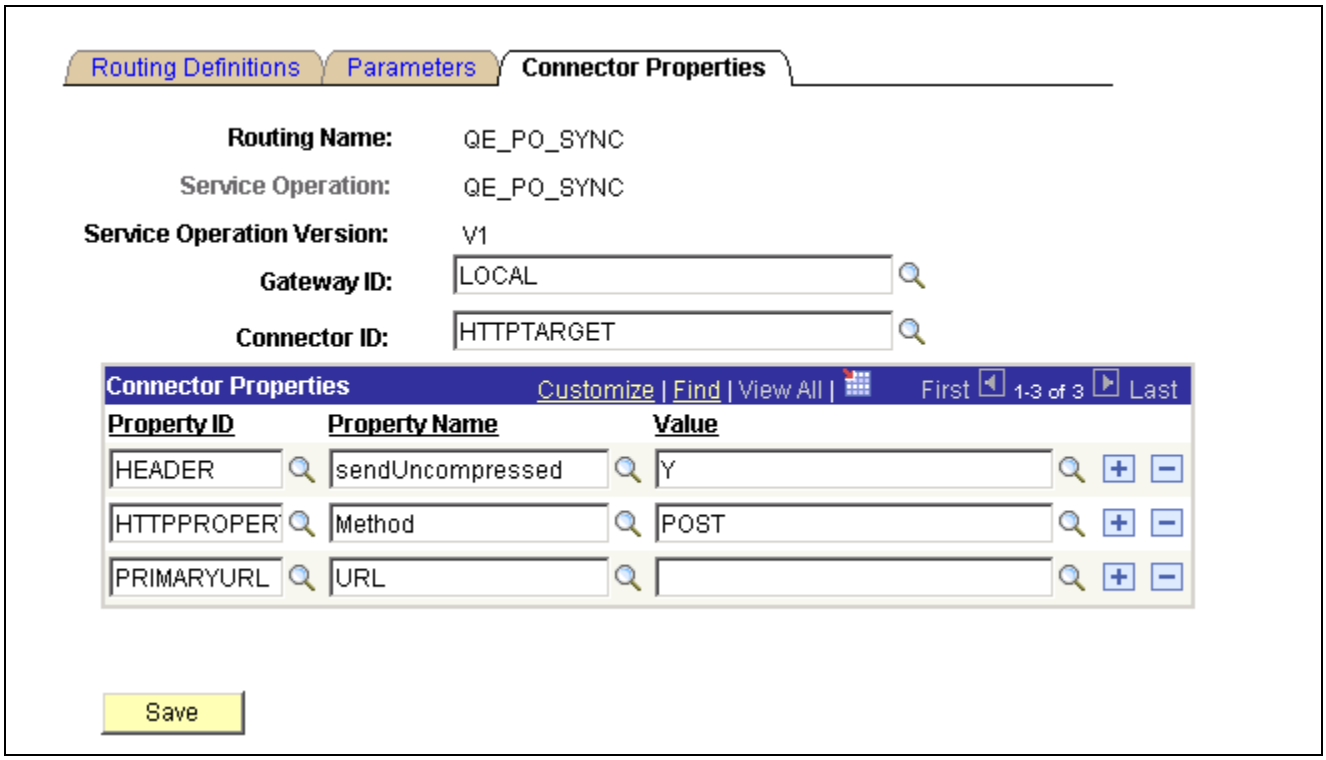
Note. The Routings-Connector Properties page displays in the Routings component only if the receiving node is not the local node.

The following graphic shows the Routings-Connector Properties pages used to define connector properties for a routing definition:



Routings-Connector Properties page

After you select an integration gateway and target connector with which to work, the system displays the required properties for the connector that you can set and override. To set or override additional properties add them to the properties list with the desired values.



Properties for the HTTP target connector

Gateway ID Click the Lookup button to select an integration gateway.

Copyright © 1988-2007, Oracle. All rights reserved.

Connector ID Click the Lookup button to select a target connector that resides on the gateway entered in the Gateway ID field.

After you select a target connector, its required properties appear.

Save Click the Save button to save your changes.

Overriding Default Integration Gateways

To override the default integration gateway, use the Gateway ID field Lookup button to select a gateway. You must also select a target connector on the new gateway to use and define properties for that connector.

Overriding Default Target Connectors

To override the default target connector on the default integration gateway, use the Gateway ID field Lookup button to select the default gateway. Use the Connector ID field Lookup button to select a different target connector that resides on the gateway and then define properties for the new connector.

Overriding Default Connector Properties

To override the default target properties for the default target connector, use the Gateway ID field Lookup button to select the default gateway. Use the Connector ID field Lookup button to select the default target connector and then adjust the gateway properties as appropriate.

Using Introspection to Create Routing Definitions

This section discusses how to:

- Select service operations for which to create routing definitions.
- Select the node to introspect.
- Select routing definitions to generate.
- View introspection results.

Understanding Using Introspection to Create Routing Definitions

You can introspect PeopleTools 8.49 nodes to create inbound or outbound point-to-point routing definitions on nodes that have matching service operations and versions for the local node. You can introspect *PIA* and *External* nodes types.

The following table lists the actions you can perform using introspection:

Routing Definition On Local Node	Routing Definition on Introspected Node	Introspection Option
Inbound point-to-point routing.	None	Delete routing on local node.
Outbound point-to-point routing.	None	Delete routing on local node.
None.	Inbound point-to-point routing.	Create outbound point-to-point routing.

Copyright © 1988-2007, Oracle. All rights reserved.

Routing Definition On Local Node	Routing Definition on Introspected Node	Introspection Option
None.	Outbound point-to-point routing.	Create inbound point-to-point routing.
None.	Any-to-local routing. (Inbound.)	Create outbound point-to-point routing.

Prerequisites for Using Introspection to Create Routing Definitions

The following prerequisites exist for using introspection to create routing definitions:

- Nodes to be introspected must be active. To verify that a node is active, open the node definition for the node and make sure that the Active box is checked on the definition.
- External nodes to be introspected must have a WSIL URL defined on the node definition.

See Also

Chapter 19, "Adding and Configuring Nodes," Configuring Nodes, page 363

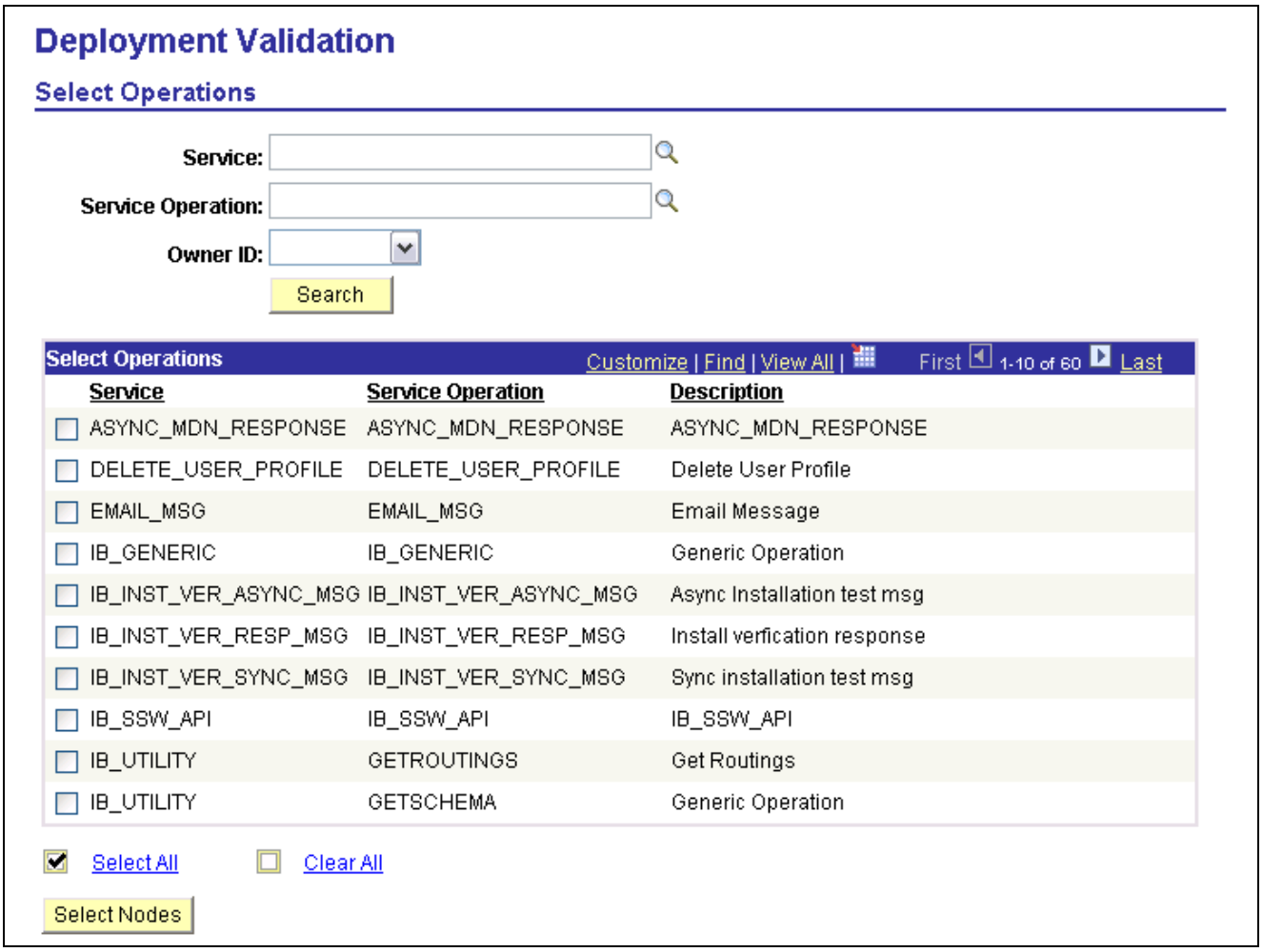
Pages Used to Using Introspection to Create Routing Definitions

Page Name	Object Name	Navigation	Usage
Deployment Validation-Select Operations	PTIB_INTROSPECT_0	Select PeopleTools, Integration Broker, Web Services, Deployment Validation.	Select the service operations for which to create routing definitions.
Introspection and Deployment Validation-Select Nodes	PTIB_INTROSPECT_1	Select PeopleTools, Integration Broker, Web Services, Deployment Validation. Click the Select Nodes button.	Select the nodes to introspect for service operation and routing data that exactly matches that on the local node.
Introspection and Deployment Validation-Introspection Results	PTIB_INTROSPECT_2	Select PeopleTools, Integration Broker, Web Services, Deployment Validation. Click the Select Nodes button. Click theIntrospect Selected Nodes button.	Select actions to perform on the node introspection results. View results of those actions performed.

Selecting Service Operations for Which to Create Routing Definitions

To begin using introspection to generate routing definitions, select the service operations for which to create routing definitions. Use the Deployment Validation-Select Operations page in the Deployment Validation component (PTIB_INTROSPECTION) shown in the following example to select service operations:

 Copyright © 1988-2007, Oracle. All rights reserved.



Deployment Validation-Select Operations page

You can search by service name and service operation. You can also search by object owner ID, if one is defined for the service. You can enter one or more of these criteria when performing your search. If you select no search options, the system searches for and returns all service operations in the database.

After you enter the search criteria and click the Search button, the results display in the Select Operations grid and you can select the service operations for which to generate routing definitions.

You can select one or more services operations for which to generate routing definitions.

To select services operations for which to generate routing definitions:

1. Select PeopleTools, Integration Broker, Web Services, Deployment Validation. The Deployment Validation-Select Operations page appears.
2. Enter search criteria for the services operations for which to generate routing definition. Provide one or more of the search criteria to narrow your search. Select no search criteria to retrieve a list of all service operations in the database.
   - In the Service field, enter a service name.
   - In the Service Operation field, enter a service operation name.
   - From the Object Owner ID dropdown list box, select the object owner of the service to provide.
3. Click the Search button.

Copyright © 1988-2007, Oracle. All rights reserved.

A Select Operations grid appears that contains the search results.

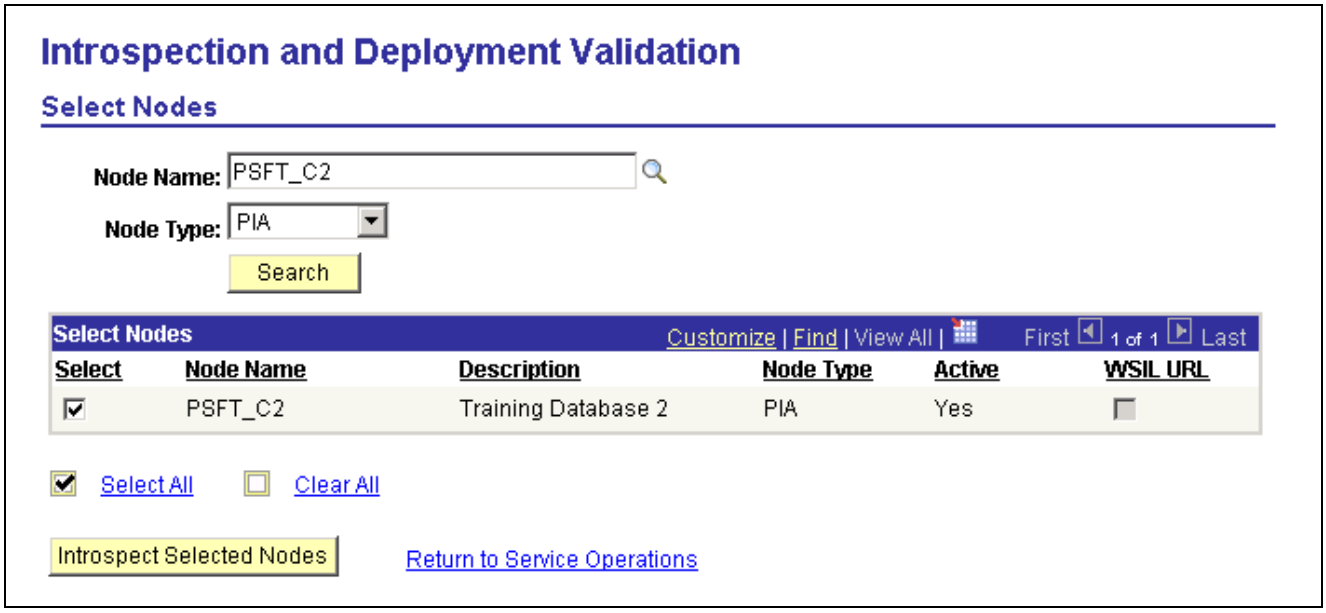
4. Check the box next to each service operation for which to generate a routing definition.

   To clear a selection, check the box again.

5. Click the Select Nodes button to proceed to select nodes to introspect.

Selecting Nodes to Introspect

Use the Introspection and Deployment Validation-Select Nodes page to select the nodes to introspect. The following example shows this page:



Introspection and Deployment Validation-Select Nodes page

To select a node to introspect:

1. Enter one or more of the following search criteria to search for a node:
   - In the Node Name field, enter a node name.
   - From the Node Type dropdown list box, select a node type. The options are:

     PIA Designates the node as a PeopleSoft database that uses PeopleSoft Integration Broker.

     External Designates the node as an entity that doesn't use PeopleSoft Integration Broker.

     Note. The ICType node type displays in the list, however you cannot introspect the ICType node type to create routing definitions.

   - Select no search criteria to retrieve a list of all nodes defined in the database.
2. Click the Search button.

   A Select Nodes grid appears that contains the search results.

3. Check the box next to the nodes to introspect.

Copyright © 1988-2007, Oracle. All rights reserved.

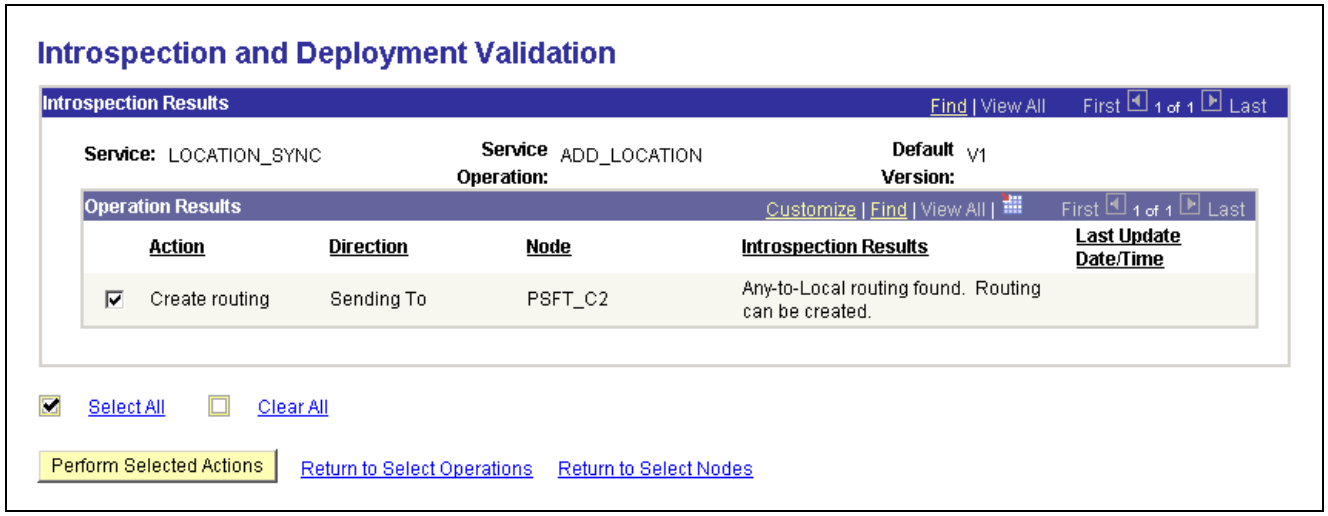
If a node displays in the list, but isn't available for selection, the check box is grayed out. A node may not be available for selection due to not being active or in the case of external nodes, no WSIL URL is defined on the node definition.

4. Click the Introspect Selected Nodes button to introspect the node or nodes that you selected.

Click the Return to Service Operations link at any time to go back to the Deployment Validation-Select Operations page to modify the selection of service operations for which to generate routing definitions.

Selecting Routing Definitions to Generate

Use the Introspection and Deployment Validation-Introspection Results page to view the introspection results and select the routing definitions to generate.



Selecting actions to perform

After you introspect one or more nodes an Operation Results box displays for each service operation for which you are generating routing definitions. Select the actions to perform and click the Perform Selected Actions button.

The following page elements appear on the Introspection and Deployment Validation-Introspection Results page:

Service The name of the service.

Service Operation The name of the service operation

Default Version The default version of the service operation

Action Indicates the possible action to perform on the service operation. The valid values are:

- *None.* Displays when a valid routing already exists between nodes. Also displays if there are no matching routing definitions on sending and receiving nodes.
- *Delete Routing.* Displays when there is a routing on the source node, but no corresponding routing on the target node.
- *Create Routing.*. Displays when matching routing definitions are present on the sending and receiving nodes.

Copyright © 1988-2007, Oracle. All rights reserved.

Direction Indicates if the direction of the transaction is inbound or outbound. The valid values are:

- *Sending To.* Indicates an outbound routing.
- *Receiving From.* Indicates an inbound routing.
- *Blank.* No routing found.

Node Indicates the name of the node introspected.

Introspection Results Indicates the results of the introspection. The valid values are:

- *A Routing exists locally. No routing found on the node. Delete local routing?*. A routing definition exists on the local database, but there is no corresponding routing on the introspected node. You may delete the routing definition on the local node.
- *No Match Found*. Indicates that no matching routing was found on the introspected node.
- *Routing Created.*The PeopleSoft system found a matching routing definition on the introspected node and created the routing.
- *Routing Deleted*. The PeopleSoft system deleted the routing.
- *Any-to-Local routing found. Routing can be created.* The target system has an any-to-local routing defined, meaning that it will accept transactions from any node. A routing will be created.
- *Point-to-point routing found. Routing can be created.*
- *Valid Routings Found*. Routing definitions between systems already exist.

Updated On Indicates the date and time that a change or delete action was performed in the current session.

Select All Click the button to select to perform all actions listed in the Action field for each service operation displayed.

Clear All Click the button to clear all selections.

Perform Selected Action Click the button to perform the action described for selected Action fields in the Operation Results grid for each service operation.

Return to Select Operations Click the link to go back to the Deployment Validation-Select Operations page to modify the selection of service operations for which to generate routing definitions.

This option displays only when introspection is initiated from the Deployment Validation component.

Return to Service Operation Click the link to go back to the Service Operations page.

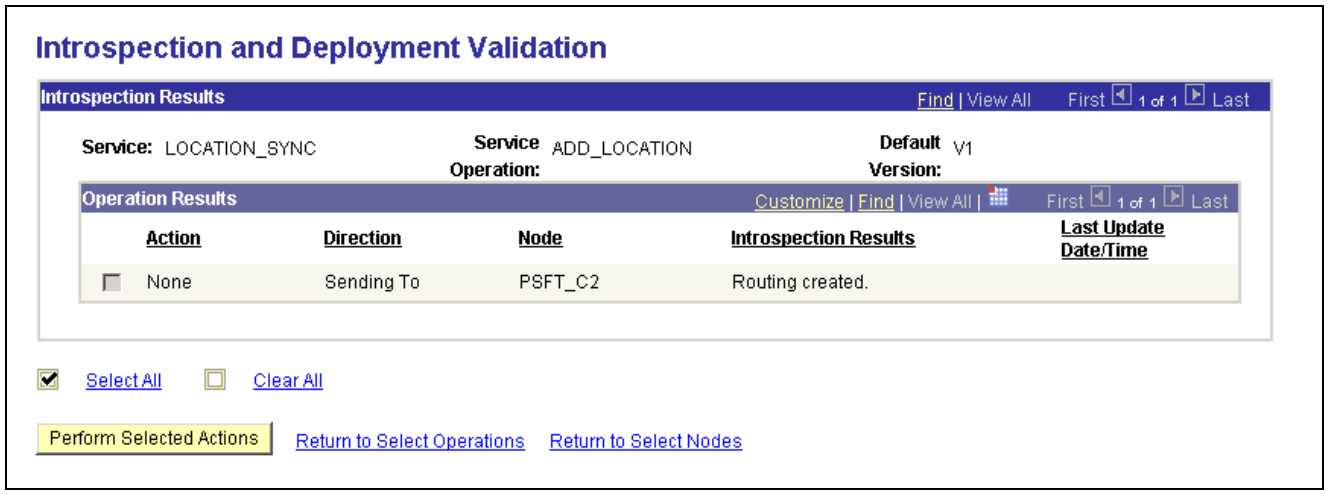
This option displays only when introspection is initiated from the Service Operations page.

Return to Select Nodes Click the link to go back to the Introspection and Deployment Validation-Select Nodes page to modify the selection of nodes to introspect.

 Copyright © 1988-2007, Oracle. All rights reserved.

View Introspection Results

After the system performs the selected actions, the following page appears.

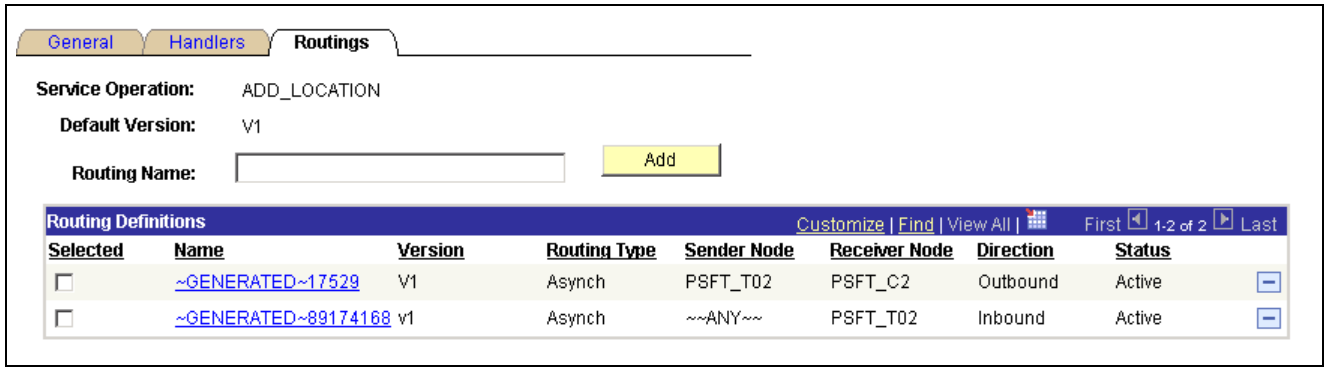


Introspection results

The Introspection Results field shows the status for each routing definition.

You can view routing definitions created using the Routing Definition page or the Service Operations-Routings page.

The following example shows the routing definitions listed for the *ADD_LOCATION* service operation on the Service Operations-Routings page:



Routing definitions for the ADD_LOCATION service operation.

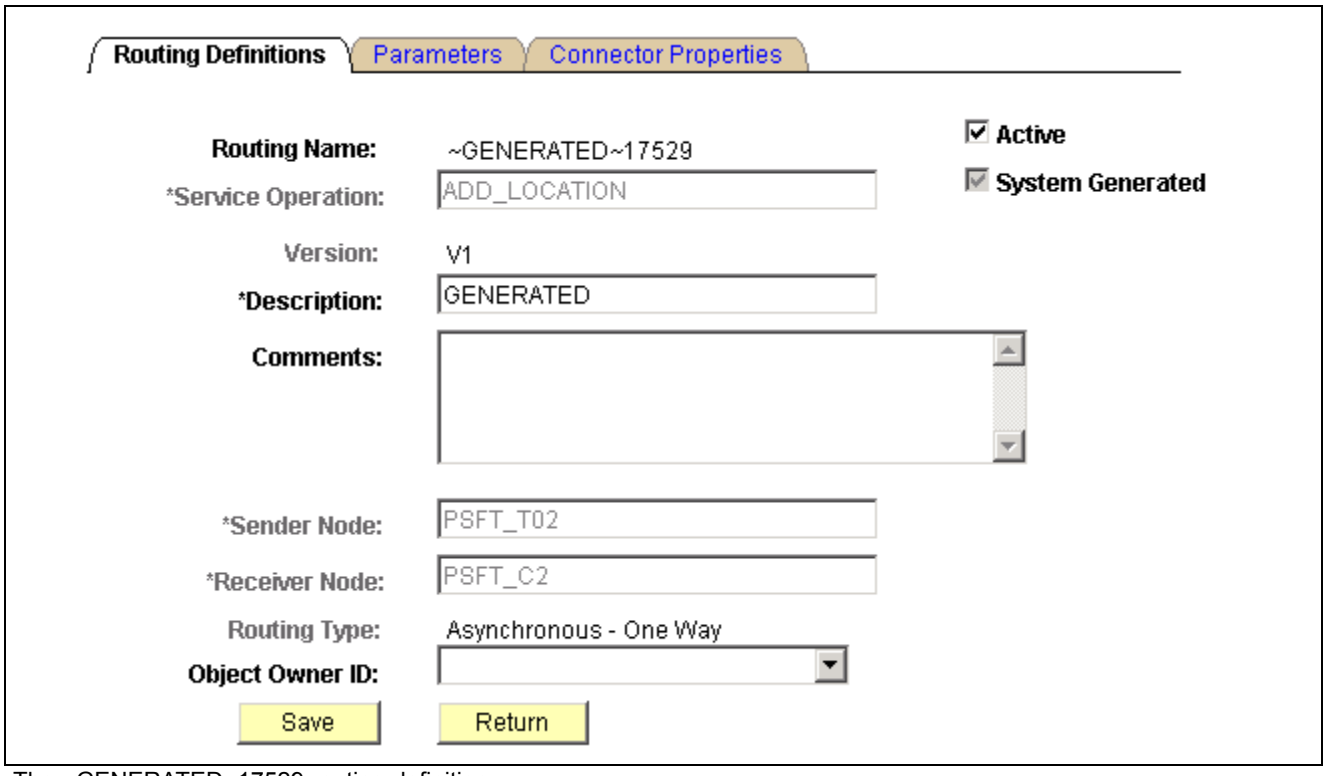
The list shows two generated routing definitions, *~GENERATED~17529* and *~GENERATED~89174168*

You can tell that the routing definition just created is *~GENERATED~17529* by the looking at the receiver node and direction.

Note. You can rename routing definitions names using the Service Administration component.

To access the routing definition, click the routing definition name. The Routing Definition page appears and displays the definition in the Routings component as shown in the following example:

Copyright © 1988-2007, Oracle. All rights reserved.



The ~GENERATED~17529 routing definition

You can view and modify the routing definition as necessary. Click the Save button to save any modifications. Click the Return button to return to the service operation record.

See Also

Chapter 18, “Managing Routing Definitions,” Defining General Routing Information, page 342

Chapter 18, “Managing Routing Definitions,” Viewing Routing Parameters for Requests and Responses, page 344

Chapter 18, “Managing Routing Definitions,” Overriding Gateway and Connector Properties, page 346

Chapter 18, “Managing Routing Definitions,” Renaming Routing Definitions, page 357

Activating and Inactivating Routing Definitions

This section discusses how to:

- Use the Routings component to activate and inactivate routing definitions.
- Use the Service Operations component to activate and inactivate routing definitions.
- Use the Nodes component to activate and inactivate routing definitions.

Understanding Activating and Inactivating Routing Definitions

You can use the Routings component or the Service Operations component to activate and inactivate routing definitions.

 Copyright © 1988-2007, Oracle. All rights reserved.

Pages Used to Activate and Inactivate Routing Definitions

Page Name	Object Name	Navigation	Usage
Routings-Routing Definitions page	IB_ROUTINGDEFN	Select PeopleTools, Integration Broker, Integration Setup, Routings.	Use the page to activate or inactivate a routing definition.
Service Operations-Routings page	IB_SERVICERTNGS	Select PeopleTools, Integration Broker, Integration Setup, Service Operations. Click the Routings tab.	Use the page to activate or inactivate a routing definition.
Nodes	IB_NODEROUTINGS	Select PeopleTools, Integration Broker, Integration Setup, Nodes. Click the Routings tab. Click the Details link for a routing definition.	Use the page to activate or inactivate a routing definition.

Activating and Inactivating Routing Definitions in the Routing Component

You can use the Routings-Routing Definitions page to activate and inactivate a routing definition.

To activate or inactivate a routing definition:

1. Select PeopleTools, Integration Broker, Integration Setup, Routings.

   Select a routing definition with which to work.
2. Perform one of the following actions:
   - To activate the routing definition, check the Active check box.
   - To inactivate the routing definition, clear the Active check box.
3. Click the Save button.

Activating and Inactivating Routing Definitions in the Service Operations Component

You can also activate and inactivate routing definitions in the Service Operations component using the Service Operations-Routings page.

See Chapter 15, “Managing Service Operations,” Activating and Inactivating Routing Definitions, page 304.

Activating and Inactivating Routing Definitions in the Nodes Component

You can use the Nodes-Routings page to access a routing definition and to activate or inactivate a routing.

To activate or inactivate a routing definition from the Nodes component:

1. Select PeopleTools, Integration Broker, Integration Setup, Nodes.
2. Click the Routings tab.

Copyright © 1988-2007, Oracle. All rights reserved.

3. Locate the routing definition to activate or inactivate and click the Details link.

   The routing definition appears in the Routing Definitions page.
4. Perform one of the following actions:
   - To activate the routing definition, check the Active check box.
   - To inactivate the routing definition, clear the Active check box.
5. Click the Save button.

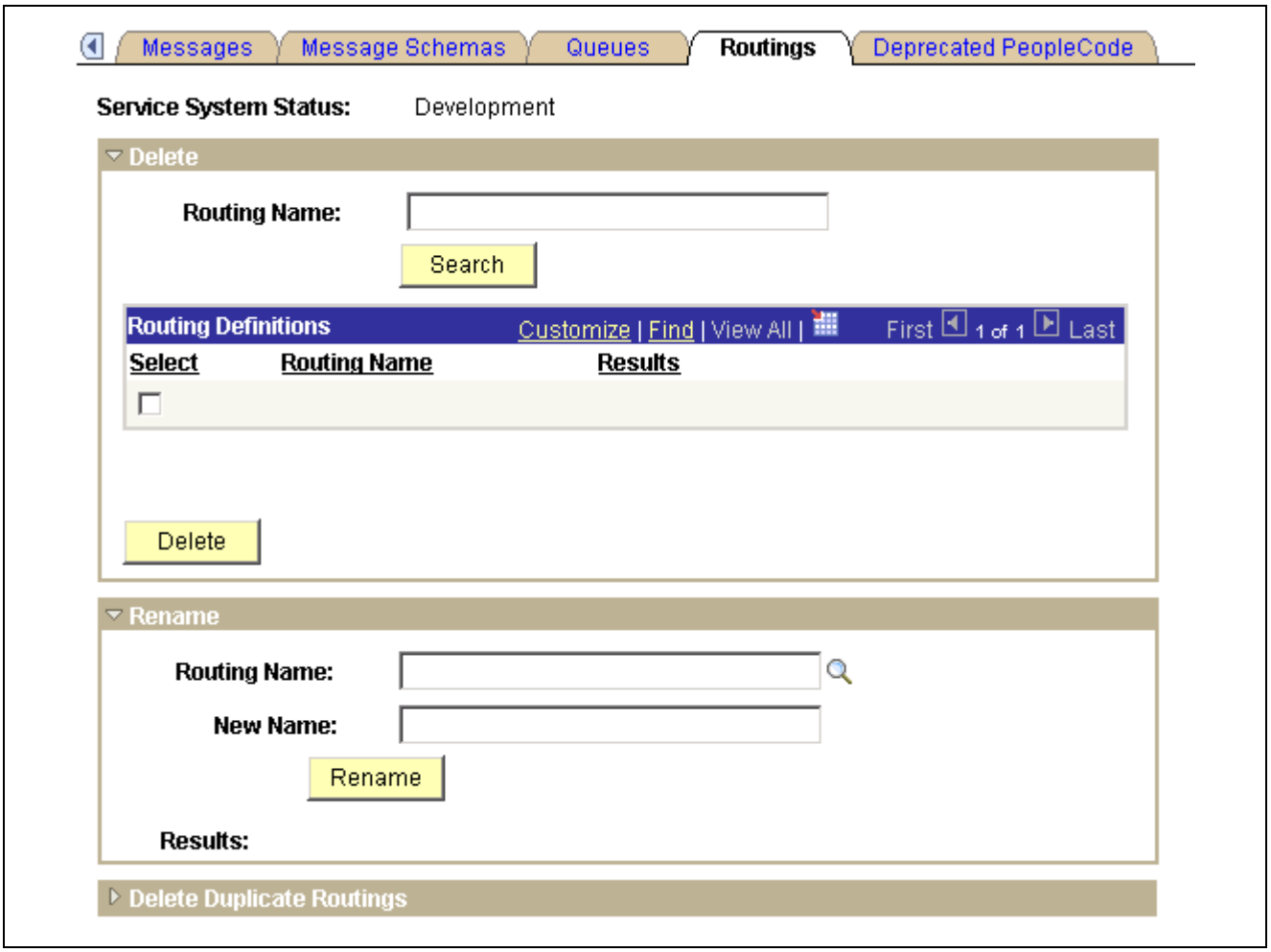
Renaming and Deleting Routing Definitions

You can rename and delete routing definitions using the Routings tab in the Service Administration component. The Routings tab contains three sections: a Delete section that enables you to delete routing definition, a Rename section that enables you to rename routing definitions, and a Delete Duplicate Routings section that enables you to view and delete duplicate routing definitions.

When you first access the Routings tab, the sections are collapsed. Click the section header arrow buttons to expand and collapse each section.

The following example shows the Routings tab with both Delete and Rename sections expanded:

 Copyright © 1988-2007, Oracle. All rights reserved.



Service Administration—Routings page

The service system status that you set on the Services Configuration page affects the ability to rename services.

This section discusses renaming and deleting routings. See the following section for information about deleting duplicate routings.

See Chapter 14, "Managing Services," Understanding Configuring PeopleSoft Integration Broker for Handling Services, page 277; Chapter 14, "Managing Services," Setting Service Configuration Properties, page 279 and Chapter 18, "Managing Routing Definitions," Deleting Duplicate Routing Definitions, page 358.

Pages Used to Rename and Delete Routing Definitions

Page Name	Object Name	Navigation	Usage
Service Administration-Routings page	IB_HOME_PAGE4	Select PeopleTools, Integration Broker, Service Utilities, Service Administration. Click the Routings tab.	Use the page to rename or delete routing definitions.

Renaming Routing Definitions

To rename a routing definition:

Copyright © 1988-2007, Oracle. All rights reserved.

1. Select PeopleTools, Integration Broker, Service Utilities. Select the Routings tab.

   The Routing page appears.
2. Click the arrow next to the Rename section header to expand the section.
3. In the Routing Name field, enter the routing definition to rename, or click the Lookup button to search for and select one.
4. In the New Name field, enter the new name for the routing definition.
5. Click the Rename button.

After you click the Rename button, the Results field displays a message that the action was successful or displays a warning or error message with a description of the problem.

Deleting Routing Definitions

To delete a routing definition:

1. Select PeopleTools, Integration Broker, Service Utilities. Select the Routing tab.

   The Routing tab appears.
2. Click the arrow next to the Delete section header to expand the section.
3. In the Routing Name field, enter the name of the routing definition to delete, and click the Search button.

   Search results display in the Routings grid.
4. Select the check box next to the routing definition or routing definitions to delete.
5. Click the Delete button.

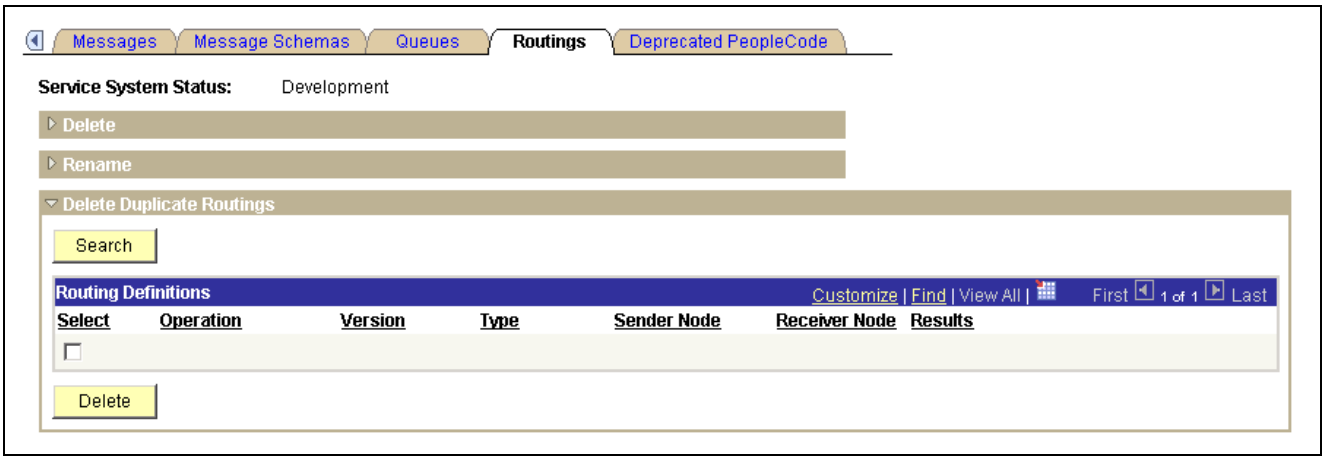
Deleting Duplicate Routing Definitions

Application upgrades and the PeopleSoft Application Designer project copy process can cause duplicate routings in the PeopleSoft system.

The Service Administration - Routings page features a Delete Duplicate Routings section that enables you to search for duplicate routings in the system and delete them. When you first access the page, all sections on the page are collapsed. Click the arrow next to the Delete Duplicate Routings section title to expand the section.

The following graphic shows the Service Administration–Routings page with the Delete Duplicate Routings section expanded:

 Copyright © 1988-2007, Oracle. All rights reserved.



Delete Duplicate Routings section of the Service Administration–Routings page.

The page displays active duplicate routings only.

Page Used to Delete Duplicate Routing Definitions

Page Name	Object Name	Navigation	Usage
Service Administration-Routings page	IB_HOME_PAGE4	Select PeopleTools, Integration Broker, Service Utilities, Service Administration. Click the Routings tab. Click the arrow next to the Delete Duplicate Routings section title to expand the section.	Use the page to delete duplicate routing definitions.

Deleting Duplicate Routings

To delete duplicate routings:

1. Select PeopleTools, Integration Broker, Service Utilities, Service Administration. Click the Routings tab.
2. Click the arrow next to the Delete Duplicate Routings section title to expand the section.
3. Click the Search button to search the system for duplicate routing definitions.

   Duplicate routing definitions populate the Routing Definitions grid and all duplicates are selected for deletion.
4. Clear the Select box for any routing definitions you do not want to delete.
5. Click the Delete button.

The Routing Definitions grid displays up to 100 routing definitions at a time. The maximum number of rows returned at a time is 1000. Use the arrow buttons to move from page to page through the search results. If the maximum number of rows is reached, an information message appears that indicates the maximum has been reached. After you delete the routing definitions, click the Search button again to return more rows.

Copyright © 1988-2007, Oracle. All rights reserved.

 Copyright © 1988-2007, Oracle. All rights reserved.

CHAPTER 19

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 Adding and Configuring 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

Prerequisites

To configure a node and its associated transactions, at least one gateway with one connector must be defined.

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)

Copyright © 1988-2007, Oracle. All rights reserved.

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.

Adding Node Definitions

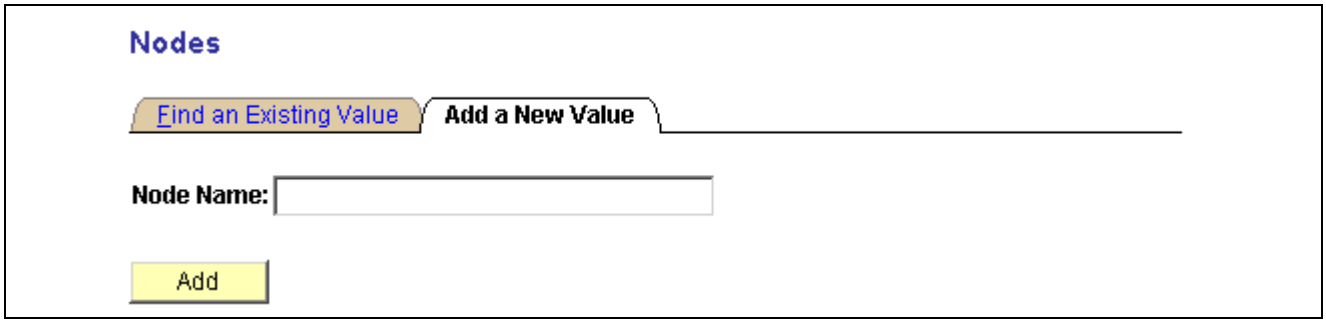
This section discusses how to add a node definition to the system.

Page Used to Add Node Definitions

Page Name	Object Name	Navigation	Usage
Nodes page	(Search)	Select PeopleTools, Integration Broker, Integration Setup, Nodes. Click the Add a New Value tab.	Add a node definition to the system.

Adding a Node Definition

The following example shows the Nodes-Add a New Value page.



Add a node definition

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.
4. Click the Add button to define the node.

   The Node Definitions tab displays.

 Copyright © 1988-2007, Oracle. All rights reserved.

Configuring Nodes

This section discusses how to:

- Define node parameters.
- Specify contact information.
- Define node properties.
- Specify node gateways and connectors.

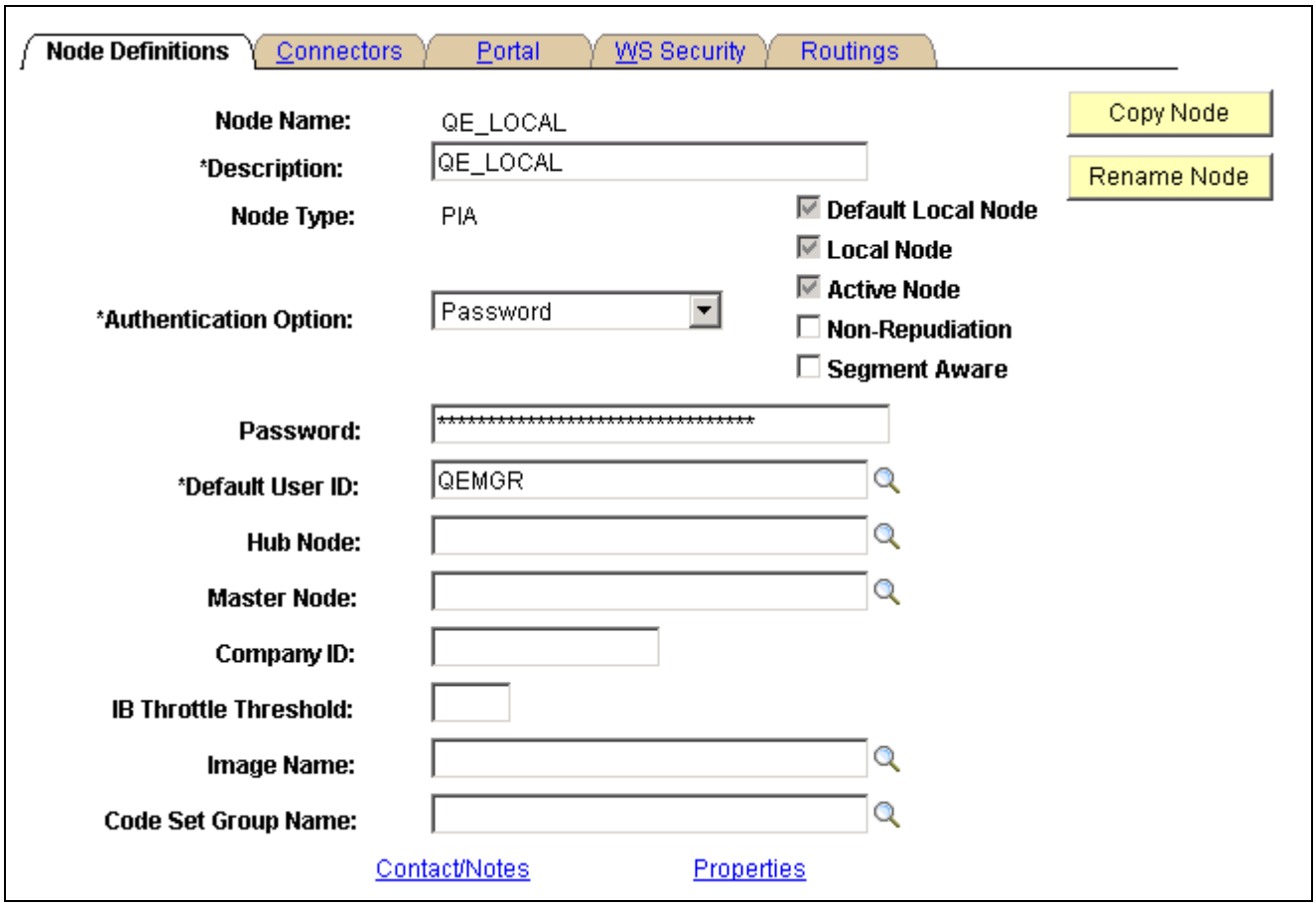
Pages Used to Configure Nodes

Page Name	Object Name	Navigation	Usage
Node Definitions page	IB_NODE	Select PeopleTools, Integration Broker, Integration Setup, Node.	Define node parameters such as its description and its default user ID. Also specify if the node is a local node, active node or is segment aware. Use this page to also enable nonrepudiation.
Node Contacts/Notes page	IB_NODECONTACT	Select PeopleTools, Integration Broker, Integration Setup, Node. Click the Contact/Notes link at the bottom of the Node Definitions 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.
Node Properties page	IB_NODEPROP	Select PeopleTools, Integration Broker, Integration Setup, Node. Click the Properties link at the bottom of the Node Definitions page.	Use the page to store additional information about the current node that can be referenced by any other node
Connectors page	IB_NODECONN	Select PeopleTools, Integration Broker, Integration Setup, Node. Click the Connectors tab.	Specify the gateway and target connector that the node uses for integrations.

Defining Node Parameters

Access the Node Definitions page.

Copyright © 1988-2007, Oracle. All rights reserved.



Node Definitions page

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.

 Copyright © 1988-2007, Oracle. All rights reserved.

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 27, "Setting Up Secure Integration Environments," Implementing Node Authentication, page 648.

Default Local Node Indicates whether the current node represents the database to which you are assigned. 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 use the Rename Node button to rename the default local node to more appropriately reflect your application or system.

Note. After you rename the default local node, you must reboot the web server.

See Chapter 19, "Adding and Configuring Nodes," Renaming or Deleting Nodes, page 370.

Local Node Indicates that the current node is either a portal node or the default local node.

Note. You can't 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.

Note. Inactivating a node will inactivate related routing definitions. You must reactivate the routing definitions manually.

See Chapter 18, "Managing Routing Definitions," Activating and Inactivating Routing Definitions, page 354.

Note. You can't inactivate the default local node.

Non-Repudiation Select to activate nonrepudiation for the current node.

Note. 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 Chapter 12, "Sending and Receiving Messages," Working With Message Segments, page 254.

Password Displays when the Authentication Option is *Password*.

Copyright © 1988-2007, Oracle. All rights reserved.

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.

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.

Note. 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 Chapter 20, "Applying Filtering, Transformation and Translation," Performing Data Translation, page 399.

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.

 Copyright © 1988-2007, Oracle. All rights reserved.

Additional information about deleting nodes is contained elsewhere in this chapter.

See Chapter 19, "Adding and Configuring Nodes," Renaming or Deleting Nodes, page 370.

Delete Node The Delete Node button displays after you have saved the initial node definition.

Note. You cannot delete the default local node.

Additional information about deleting nodes is contained elsewhere in this chapter.

See Chapter 19, "Adding and Configuring Nodes," Renaming or Deleting Nodes, page 370.

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.

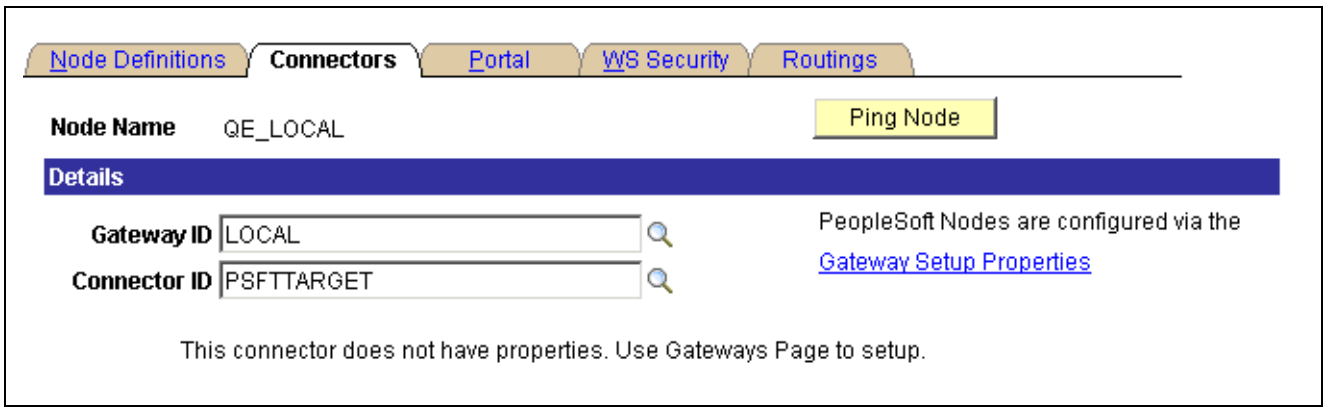
Search: The property is used for searching.

Property Name Enter a new property name or select an existing property of the selected name type.

Copyright © 1988-2007, Oracle. All rights reserved.

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.

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. 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. 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 7, "Managing Integration Gateways," Editing Connector Properties, page 58.

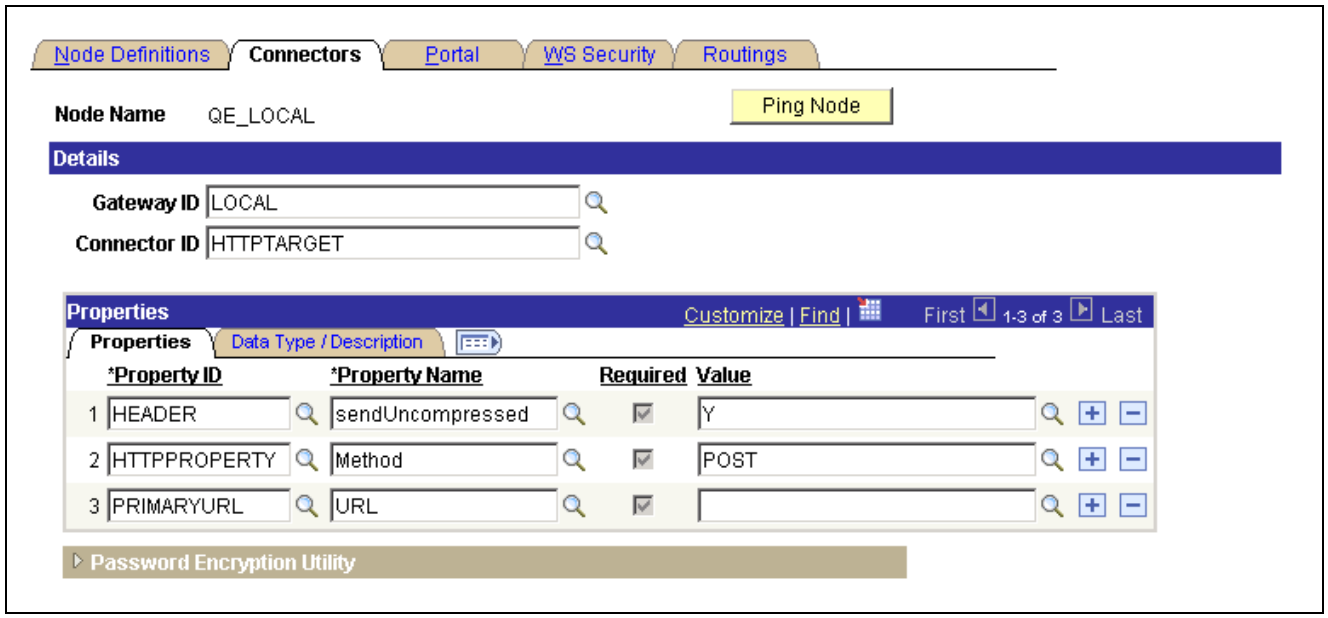
 Copyright © 1988-2007, Oracle. All rights reserved.

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.



HTTP target connector properties

Changes you make on this page have no effect on the originals. You can:

- 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.

Copyright © 1988-2007, Oracle. All rights reserved.

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 7, “Managing Integration Gateways,” Encrypting Passwords, page 66

Chapter 7, “Managing Integration Gateways,” Editing Connector Properties, page 58

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.

 Copyright © 1988-2007, Oracle. All rights reserved.

Page Used to Rename and Delete Nodes

Page Name	Object Name	Navigation	Usage
Domain Status page	AMM_MULTIDOM	SelectPeopleTools, Integration Broker, Service Operations Monitor, Administration, Domain Status.	Activate and deactivate domains in the messaging system.
Asynchronous Details page	IB_MONITOR_DET	SelectPeopleTools, Integration Broker, Service Operations Monitor, Monitor, Asynchronous Details.	Archive asynchronous service operations one at a time.
Synchronous Details page	IB_MONITOR_DET	SelectPeopleTools, Integration Broker, Service Operations Monitor, Monitor, Synchronous Details.	Archive synchronous service operations one at a time.
Run Archive	RUN_APMSGARCH	SelectPeopleTools, Integration Broker, Service Operations Monitor, Monitor, Archive Monitor Data.	Batch archive service operations.
Node Definitions	IB_NODE	SelectPeopleTools, Integration Broker, Integration Setup, Nodes.	Rename or delete a node.

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 dropdown 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.

Copyright © 1988-2007, Oracle. All rights reserved.

- 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.
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

Chapter 21, "Using the Service Operations Monitor," Running Batch Service Operation Archiving Processes, page 455

Chapter 21, "Using the Service Operations Monitor," Purging Runtime Monitor Tables, page 493

Enterprise PeopleTools 8.49 PeopleBook: Data Management, "Using PeopleSoft Data Mover," Understanding Data Mover Scripts

 Copyright © 1988-2007, Oracle. All rights reserved.

CHAPTER 20

Applying Filtering, Transformation and Translation

This chapter discusses how to:

- Develop transformation programs.
- Filter messages and generate errors.
- Apply transformations.
- Perform data translation.
- Use the Oracle XSL Mapper to develop transformation programs

Understanding Filtering, Transformation, and Translation

Filtering, transformation, and translation are all accomplished by applying an Application Engine *transform program* to an outbound or inbound message. You can use transform programs to:

- *Filter* a message based on its content, to determine whether to pass it through to its destination or to subsequent steps of the transform program.
- Perform *transformation* on a message to make its structure comply with the receiving system's requirements.
- Perform *data translation* on a message so that its data is represented according to the receiving system's conventions.

  Simple translation might be required if the two systems use different values to represent the same information for a given field.

  Complex translation might involve augmenting or replacing groups of fields with a completely different structure and encoding.

If your PeopleSoft application uses the PeopleCode XmlDoc or SoapDoc classes to generate or process a message, the message probably doesn't adhere to the PeopleSoft rowset-based message format.

See Also

Chapter 8, "Understanding Supported Message Structures," page 81

Understanding Transform Programs

This section provides overview information about transform programs

Copyright © 1988-2007, Oracle. All rights reserved.

Transform Programs

A transform program is a type of PeopleSoft Application Engine program. After you create a new transform application engine program, you add steps and actions to the program, and then add code to the steps and actions that performs data transformation, filtering or translation.

To develop a transform program, you must know the initial structure and possibly the content of the message with which you are working, as well as the structure (and content) of the result you want to achieve. Make sure that all participating nodes agree on a format or employ transformations to accommodate the variations from node to node.

The message data is made available to your transform program in a PeopleCode system variable after being extracted from the wrapper in which it was transmitted. The format of this wrapper depends on the transmission method, but is irrelevant to the transform program.

Filtering, transformation, or translation can be necessary for messages sent between two PeopleSoft Integration Broker nodes, or between a PeopleSoft Integration Broker node and a third-party application. Any participating node with PeopleSoft Integration Broker installed — the source, the target, or a hub — can apply a transform program to a given message.

You specify which transform program to apply within a routing definition for a service operation.

Note. With PeopleSoft Integration Broker, the term *node* refers to a system or application participating in an integration, but in this chapter a *node* is also a structural element in an XML document. The context in which the term is used should make its meaning clear.

Transform programs cannot modify the following messaging features:

- Transmission protocols.

  You handle a given protocol by selecting an appropriate target connector for the target node's local gateway, or by directing a third-party sender to the appropriate listening connector on the default local node's local gateway. You can select from the delivered connectors or develop new ones.

- Character set encoding.

  This is handled by the PeopleSoft globalization system.

Transformation Programming Languages

You can use PeopleCode or Extensible Stylesheet Language Transformation (XSLT) as programming languages for creating transformation logic.

XSLT is a well-recognized standard language perfectly suited to manipulating XML structures, so it's highly recommended for transformations. Because of its straightforward template-based approach to accessing the codeset repository, XSLT is also recommended for data translation.

Note. When programming using XSLT, you can hand-code the XSLT or use the Oracle XSL Mapper to graphically associate records and fields. The Oracle XSL Mapper then automatically generates the XSLT code.

However, you cannot use XSLT to filter messages based on content, so filtering must be implemented in PeopleCode.

 Copyright © 1988-2007, Oracle. All rights reserved.

Note. Filtering must be implemented using PeopleCode

You can use both XSLT and PeopleCode steps in a single transform program.

Each XSLT program must be enclosed in the following wrapper:

Note. When using Oracle XSL Mapper, the mapper automatically encloses the program in this wrapper.

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

Third-party XSLT development tools may generate a wrapper that specifies a different URI. Make sure the URI in your program is exactly as shown here, or your program may not work as expected.

You can find more information about XSLT at the World Wide Web Consortium (W3C) web site.

Third-Party Considerations

When no transformation is applied, applications using PeopleSoft Integration Broker send, and expect to receive, messages containing data that conforms to a minimum XML message structure: the PeopleSoft rowset-based message format.

When exchanging messages with third-party applications, you can:

- Employ a transformation at the PeopleSoft end of each transaction to convert messages to or from the PeopleSoft rowset-based message format.
- Require third-party applications to send and receive messages that comply with the PeopleSoft rowset-based message format. Third-party applications must comply with the rowset-based message format if *both* of the following are true:
  - Your PeopleSoft application uses the PeopleCode Message and Rowset classes to generate and send, or receive and process messages with Integration Broker.
  - You don't want to employ PeopleSoft Integration Broker transformations to accommodate the third-party application.

Note. Third parties can submit messages to PeopleSoft Integration Broker systems using any listening connector registered with the local gateway. Regardless of the message data format, the third-party system is responsible for properly constructing the wrapper that contains the message data, and using the appropriate tools and protocols to address the connector.

Defining Transform Programs

This section discusses how to define transform programs.

Copyright © 1988-2007, Oracle. All rights reserved.

Understanding Defining Transform Programs

This section contains information about defining transform programs

Transform Program Type

To create a transform program, in the Program Properties dialog box for the application engine program you must specify that the program type is *Transform Only*. After you select this option, input message name, output message name, input root element and output root element fields display.

Input and Output Message Names

When developing a transformation program, PeopleSoft Integration Broker enables you to specify the schema of the message that is going into the transform (input message and version), as well as the schema of the message that is the output of the program (output message and version).

Note. You must specify this information when using the Oracle XSL Mapper to develop transformation programs.

These fields are required when developing transformations using the Oracle XSL Mapper.

In all other cases, these fields are optional, since there may be occasions where a transformation is general in nature and used by many messages. For example, it might be a transform that changes certain fields regardless of the message shape. In cases such as these, you would not want to define a specific input or output shape, since the transform program is only changing fields.

Input and Output Root Elements

When working with nonrowset-based messages, there may be situations where the schemas for input and output messages have multiple root elements. However, Oracle XSL Mapper uses only one of the root elements on the input side as well as only one on the output side.

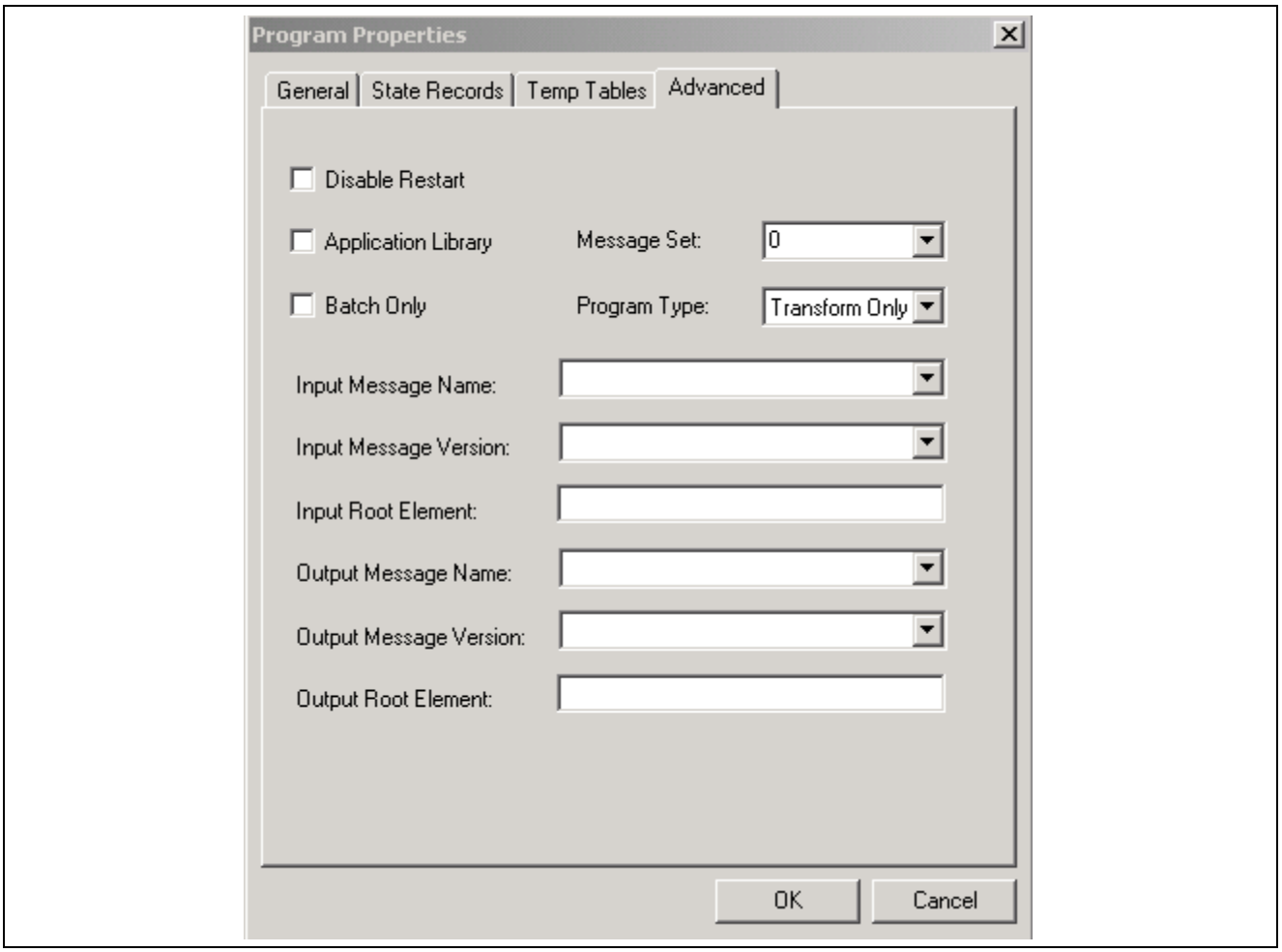
When using Oracle XSL Mapper to build XSLT transformations, you may specify the input and output root elements in the Program Properties dialog box. If you do not specify an input or output root element, Oracle XSL Mapper uses the first root element in the schema.

Defining a Transform Program

To define a transform program, create a new application engine object in PeopleSoft Application Designer. Then in the Program Properties dialog box you specify the program type as a transform program.

The following graphic shows the Program Properties dialog box for an application engine program. Note that the Program Type field displays *Transform Only*.

Copyright © 1988-2007, Oracle. All rights reserved.



Program Properties — Advanced tab

To define a transform program:

1. In PeopleSoft Application Designer, select File, New, App Engine Program and click the OK button.

   A new application engine program window appears.

2. On the toolbar, click the *Properties* button.

   The Program Properties dialog box appears.

3. Click the Advanced tab.

4. From the Program Type dropdown list box, select *Transform Only*.

   Additional fields relating to input messages, output messages and root elements appear.

5. Select an input message and version:

   a. From the Input Message Name dropdown list box, select the name of the message before transformation is applied.

   b. From the Input Message Version dropdown list box, select the version of the input message.

6. In the Input Root Element field, enter the name of the input schema root element to use.

   Enter a value in this field if the input message has multiple root elements. If the input message has multiple root elements and you do not enter an input root element, the first root element in the message is used for transformation.

Copyright © 1988-2007, Oracle. All rights reserved.

This field is disabled when the input message is a rowset-based message.

7. Select an output message and version.
    a. From the Output Message Name dropdown list box, select the name of the message after transformation is applied.
    b. From the Output Message Version dropdown list box, select the version of the output message.
8. In the Output Root Element field, enter the name of the output schema root element to use.

    This field is disabled when working with rowset-based messages.
9. Click the OK button.
10. The Program Properties dialog box closes.
11. Select File, Save.

Developing Transform Programs

This section discusses how to:

- Insert steps and actions into transform programs.
- Work with transform programs.
- Access message data.
- Make working data available globally.
- Preserve record and field aliases.

Understanding Developing Transform Programs

Following are some points to keep in mind when working with transform programs:

- Each transform program step operates on the message content that results from the previous step, so you can break your transform program into a sequence of discrete steps.
- Multiple transform actions within a step can produce unwanted effects, so insert each XSLT or PeopleCode action in its own step.
- XSLT works only on XML DOM-compliant data, so PeopleSoft Integration Broker assures that both outbound and inbound messages are in XML DOM-compliant form when transform programs are applied to them.
- XSLT is not supported by PeopleSoft on the OS/390 or z/OS operating systems. Transformations must use PeopleCode on these platforms.

A transformation can modify an entire message until it no longer resembles the original. So can a data translation. In a transformation, you must hard-code what you want to accomplish, whereas the data translation relies on a repository of *codeset metadata* that you define. This means you can establish consistent rule-based translations and reuse the same translation rules without having to reenter them.

Although you can combine transformation and data translation in a single transform step, it's better to keep these processes in separate steps and produce a modular program that you can more easily maintain, with code you can reuse in other transform programs.

 Copyright © 1988-2007, Oracle. All rights reserved.

Inserting Steps and Actions into Transform Programs

This section describes how to insert steps and actions into transform programs.

Understanding Inserting Steps and Actions

After you define a transform program, you insert steps and actions as you would with any other application engine program to construct the transformation program.

The two types of actions you can add to steps when building a transform program are *XSLT* and *PeopleCode.*

Inserting XSLT Actions

When you select *XSLT* as the step action type you can develop the transform code using Oracle XSL Mapper or you can hand-code the program.

Installing, configuring and using Oracle XSL Mapper to develop transform programs is discussed elsewhere in this chapter.

See Chapter 20, "Applying Filtering, Transformation and Translation," Developing Transformations Using Oracle XSL Mapper, page 384.

Note. After selecting XSLT as the action type, you must save the program before you can choose to use Oracle XSL Mapper or hand-code the program.

To insert XSLT actions:

1. From the Action Type dropdown list, select *XSLT*.
2. (Optional.) In the XSLT Description field, enter a description for the XSLT action.
3. Save the transform program.

   A Graphical Mapper dropdown list appears.
4. Choose how to code the XSLT action:
   - To use Oracle XSL Mapper , click the XSLT action to highlight it. Then double-click the action to launch the mapper tool.
   - To hand-code the program, from the Graphic Mapper dropdown list box, select *No*. Right-click the action and select View XSLT.
5. Create code for the step/action.
   - If using Oracle XSL Mapper, beginning mapping records and fields as appropriate.
   - If hand-coding using XSLT, right-click the action and select View XSLT. The programming window appears and you can begin coding.

Inserting PeopleCode Actions

To insert PeopleCode actions:

1. From the Action Type dropdown list, select *PeopleCode*.
2. (Optional.) In the PeopleCode Description field, enter a description for the PeopleCode action.
3. Save the program.
4. Right-click and select View PeopleCode to open the programming window.
5. Enter PeopleCode appropriate for the step and action.

Copyright © 1988-2007, Oracle. All rights reserved.

Invoking Transform Programs

Your transform program is invoked by PeopleSoft Integration Broker if you specify its name in the a routing definition for a service operation.

Renaming or Deleting Transform Programs

To invoke a transform program, you specify it as part of a routing definition. If you subsequently rename or delete that transform program, it still appears in the routing definition, so service operations using that modifier will fail. To prevent this, do the following:

- If you rename a transform program, make sure you reselect it by its new name in any routing definitions that apply it.
- If you delete a transform program, make sure you select a different program (or no program) in any routing definitions where that apply it.

Tracing Transform Programs

For debugging purposes, you can trigger a trace of your transform program by adding a specific value to the Application Engine *trace* parameter, in one of the following ways:

- Specify the TRACE switch on the Application Engine command line, with the value *8192* added, for example:

```
-TRACE 8192
```

- Add the value *8192* to the TRACEAE parameter in the appropriate application server or Process Scheduler server configuration file, for example:

```
TRACEAE=8192
```

See Also

Enterprise PeopleTools 8.49 PeopleBook: PeopleCode API Reference, "XmlDoc Class"

Enterprise PeopleTools 8.49 PeopleBook: PeopleSoft Application Engine, "Tracing Application Engine Programs"

Enterprise PeopleTools 8.49 PeopleBook: PeopleSoft Process Scheduler, "Using the PSADMIN Utility"

http://www.w3.org/Style/XSL/

Accessing Message Data

When PeopleSoft Integration Broker invokes a transform program, it inserts the message content into a PeopleCode system variable, *%TransformData*, which remains in scope throughout the program. Each step can access the variable in turn and modify its content, which then becomes available to the next step.

XSLT and PeopleCode steps access %TransformData differently:

- In XSLT, the data is automatically made available to your program. The XSLT program is literally a presentation of the output structure and data, which includes *xsl*tags that reference, process and incorporate the input data into the output structure. There's no need to explicitly refer to %TransformData, which automatically receives the interpreted result of the XSLT.
- In PeopleCode, use the PeopleCode *TransformData* class to access %TransformData. You then access the XML data as a property of the TransformData object called *XmlDoc,* which you assign to an XmlDoc object

 Copyright © 1988-2007, Oracle. All rights reserved.

and process normally. Because the XmlDoc object is a reference to the data portion of %TransformData, your modifications are automatically passed back to the system variable.

Using the TransformData Class

The PeopleCode TransformData class has several properties:

Property	Description
XmlDoc	Contains the XML message data. You can assign this to an XmlDoc object and process the data using the XmlDoc class methods and properties. This property is read/write.
Status	Communicates the success or failure of the transform program step to PeopleSoft Integration Broker. Set to *0* for success, the default value. Set to *1* to indicate that the message failed a filtering step. Set to *2* to indicate an error occurred. This property is read/write. See Chapter 20, "Applying Filtering, Transformation and Translation," Filtering Messages, page 394.
SourceNode	The name of the node sending the message. This property is read only.
DestNode	The name of the node receiving the message. This property is read only.
SourceMsgName	The name of the message at the sending node. This property is read only.
DestMsgName	The name of the message at the receiving node. This property is read only.
SourceMsgVersion	The name of the message version at the sending node. This property is read only.
DestMsgVersion	The name of the message version at the receiving node. This property is read only.

Note. Because transform programs can apply to both request and synchronous response messages, the node *sending* the message could be a synchronous transaction target node that's sending a response back to the synchronous transaction source node, which in this case is the *receiving* node.

Handling Non-XML Data

Because they work only with XML DOM-compliant data, neither XSLT nor PeopleCode transform steps can process non-XML data. The XML DOM provides a way to incorporate such data into an XML structure so your transform programs won't produce errors.

If you're generating a non-XML outbound message in your PeopleSoft application, it's up to you to insert your message content into a special element containing a CDATA tag, as follows:

```
<any_tag psnonxml="yes">
   <![CDATA[your_nonXML_message_content]]>
```

Copyright © 1988-2007, Oracle. All rights reserved.

```
</any_tag>
```

Note. *Any_tag* can be any tag you want to use.

The following restrictions apply to the content of inbound non-XML messages, such as those in CSV or PDF format, sent by third-party applications:

- Inbound non-XML text messages must be encoded as UTF-8-compliant characters.
- Inbound non-text, or binary, messages must be encoded in base64 format.

Making Working Storage Data Available Globally

XSLT transform steps can't access external data, but PeopleCode can. XSLT also has no global variables. However, the message itself is global, and can be used to pass working or external data to all steps in the transform program. During a PeopleCode step, you can add a special node to the message to contain the data, which is then available to subsequent transform steps.

Following is an example of a minimal input message:

```
<?xml version="1.0"?>
<Header>
  <LANGUAGE_CODE>en_us</LANGUAGE_CODE>
  <STATUS_CODE>1000</STATUS_CODE>
</Header>
```

The following PeopleCode inserts a node in the message to contain working data, by convention called *psft_workingstorage.* Then the PeopleCode inserts the current system date into that node:

```
/* Get the data from the AE Runtime */
Local TransformData &incomingData = %TransformData;
/* Set a temp object to contain the incoming document */
Local XmlDoc &inputDoc = &incomingData.XmlDoc;
/* Add a working storage node*/
Local XmlNode &wrkStorageNode =
 &inputDoc.DocumentElement.AddElement("psft_workingstorage");
/* Add the current system date to the working storage*/
Local XmlNode &sysDateNode = &wrkStorageNode.AddElement("sysdate");
&sysDateNode.NodeValue = String(%Date);
```

Following is the resulting output message:

```
<?xml version="1.0"?>
<Header>
  <LANGUAGE_CODE>en_us</LANGUAGE_CODE>
  <STATUS_CODE>0</STATUS_CODE>
  <psft_workingstorage>
    <sysdate>2002-01-24</sysdate>
  </psft_workingstorage>
</Header>
```

Any subsequent transform step now has access to the current system date. Make sure the last step that uses the *psft_workingstorage* node removes it from the final output, as with this XSLT fragment:

```
<xsl:template match="psft_workingstorage">
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
   <!-- Do not copy this node -->
</xsl:template>
```

Preserving Record and Field Aliases

When you apply a transform program to a rowset-based message, Integration Broker submits the message to the program in its final XML DOM compliant form, with any aliases you defined in place of the corresponding original record and field names.

In a PeopleCode transformation, the message is initially available as an XmlDoc object. However, you may want to transform the message using the PeopleCode Rowset class. Because XmlDoc object structure is compatible with Rowset object structure, you can copy the XML data to a rowset using the XmlDoc class CopyToRowset method.

Because the rowset to which you copy the data must be based on a message object instantiated from your original message, the rowset uses the message's original record and field names. If you defined aliases for any of the message records or fields, you must ensure that the rowset uses those aliases instead, or the XmlDoc data won't copy successfully.

The following set of conditions summarizes this situation:

- The message definition includes at least one record or field alias.
- You're applying a transform program to the message.
- Your transform program includes a PeopleCode step.
- The PeopleCode step uses a Rowset object to hold the message data.

Using Optional CopyToRowset Parameters

To make sure the rowset object uses the record and field aliases that exist in the XML data, you must specify two optional string parameters in the CopyToRowset method, which convey the message name and version:

```
CopyToRowset(&Rowset, Message_Name, Version)
```

The integration engine uses any aliases it finds in the specified message definition to rename the appropriate records and fields in the rowset object before copying the data. Following is an example of a rowset-based transform step that preserves aliases:

```
Local Message &TempMSG;
Local Rowset &TempRS;

/* Get the data from the AE Runtime */
Local TransformData &tempData = %TransformData;
/* Set a temp object to contain the incoming document */
Local XmlDoc &tempDoc = &tempData.XmlDoc;

/* Create a working rowset (no aliases used) */
&TempMSG = CreateMessage(Message.MyMsgName);
&TempRS = &TempMSG.GetRowset();

/* Copy message data to rowset (restoring aliases) */
&OK = &tempDoc.CopyToRowset(&TempRS, "MY_MSG_NAME", "MY_MSG_VERSION");

   /* . . .Transform rowset data. . . */
```

Copyright © 1988-2007, Oracle. All rights reserved.

```
/* Copy transformed rowset back to XmlDoc object */
&OK = &tempDoc.CopyRowset(&TempRS, "MY_MSG_NAME", "MY_MSG_VERSION");
```

See Also

Enterprise PeopleTools 8.49 PeopleBook: PeopleCode API Reference, "XmlDoc Class," XmlDoc Methods

Developing Transformations Using Oracle XSL Mapper

This section discusses how to develop transformations using Oracle XSL Mapper.

Understanding Oracle XSL Mapper

Oracle USA has developed a tool that enables you to graphically map records and fields and that creates the underlying XSLT transformation code for you. You launch this tool directly from a PeopleSoft application engine transformation program. When you save the XSLT code in Oracle XSL Mapper, it automatically gets saved to the application database and your application engine transform program.

Note. You cannot use Oracle XSL Mapper to modify XSLT that you've hand-coded or created with any other XSLT editing tool.

Oracle XSL Mapper is an Oracle JDeveloper 10g plug-in. As of the release date for PeopleTools 8.48, Oracle JDeveloper 10g Release 2 (10.1.2.0.2) is the version that PeopleSoft supports.

Development Considerations

Note the following as you develop transformations using Oracle XSL Mapper:

- When you save XSL maps that you create in the mapper, the underlying XSL code is automatically saved to the application engine program.
- The mapper does not support codesets and working storage constructs. You must add these constructs manually into the XSL code using the Source view of the mapper.

Prerequisites

To use Oracle XSL Mapper you must:

- Install Oracle BPEL Designer.

  Oracle XSL Mapper is part of Oracle JDeveloper that comes as part Oracle BPEL Designer. You get Oracle BPEL Designer with the download of Oracle BPEL Process Manager 10.12.0.2.

- In PeopleSoft Configuration Manager, specify the path to the Oracle JDeveloper 10g installation location.
- All messages used in the mapper must have schemas generated for them.

  For rowset-based messages, use the Message Definitions–Schema page to generate schemas. For nonrowset-based messages, use the Message Definitions–Schema page to add or import schemas for these types of messages.

- You must create a Transform Only application engine program and define the program properties described earlier in this chapter.

 Copyright © 1988-2007, Oracle. All rights reserved.

Installing Oracle XSL Mapper

Oracle XSL Mapper version 10.1.2.0.2 is a tool that provides the ability to rapidly create XSL-based transformation maps to support integration of PeopleSoft to third-party applications deployed through Integration Broker. The Oracle XSL Mapper is a component of Oracle BPEL Designer and is included in Oracle JDeveloper 10g. Release 2 (10.1.2.0.2). Access to this feature requires the download of Oracle Jdeveloper 10g Release 2 (10.1.2) at the location below.

See http://www.oracle.com/technology/index.html

Specifying the Path to the Oracle XSL Mapper Installation Location

To configure Oracle XSL Mapper you must specify the path to the installation location of Oracle JDeveloper on your system in PeopleSoft Configuration Manager.

To specify the path to the Oracle XSL Mapper installation location:

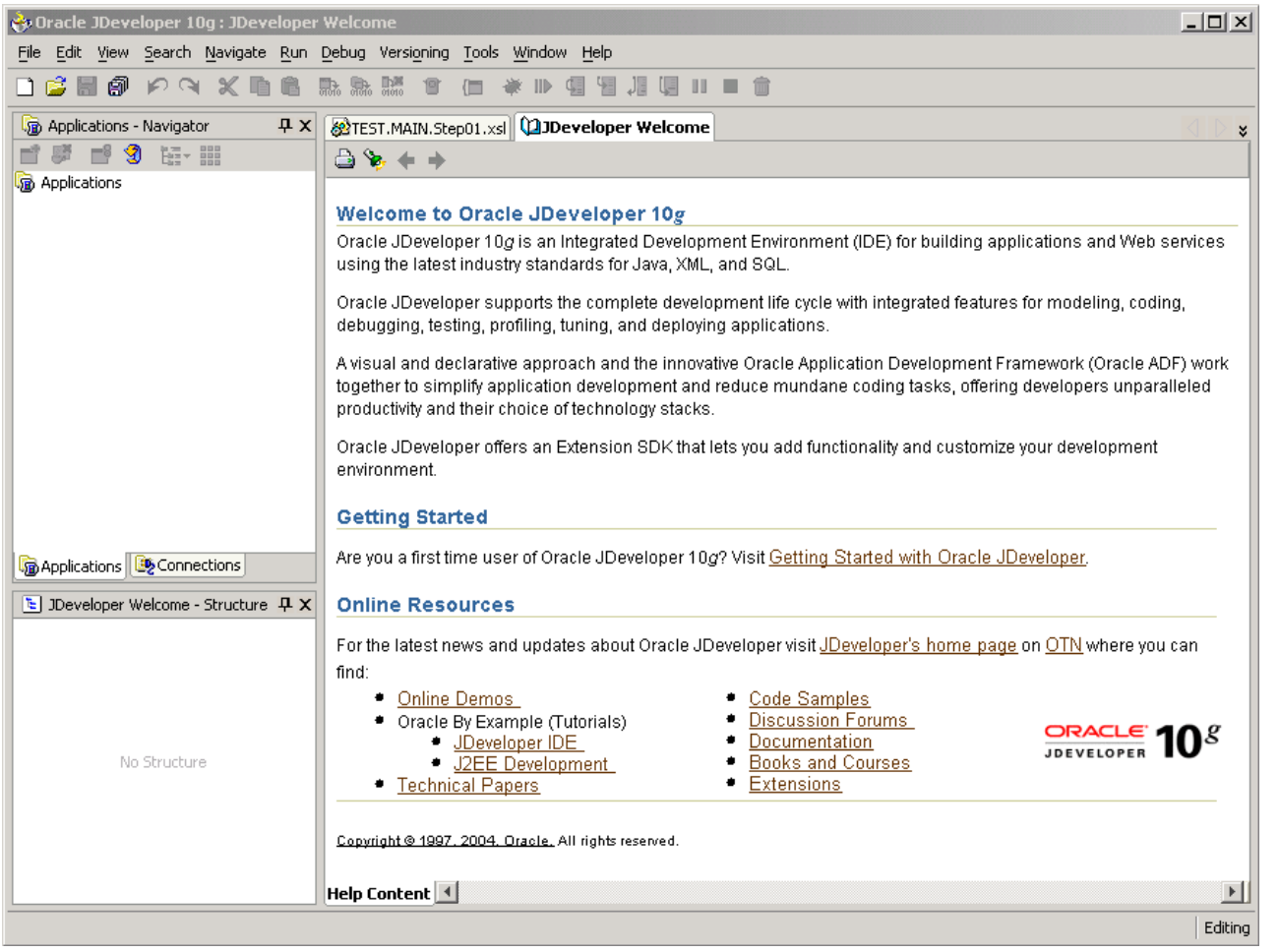
1. Open PeopleSoft Configuration Manager (pscfg.exe).
2. Click the Crystal/Bus. Interlink/JDeveloper tab.
3. Locate the JDeveloper Home Directory section at the bottom of the page.
4. In the JDeveloper Home field, enter the path or browse to the location where JDeveloper is installed.
5. Click the Apply button.
6. Click the OK button.

Launching Oracle XSL Mapper

You launch Oracle XSL Mapper from within an application engine transform program.

The first time you launch the mapper, the following Oracle JDeveloper 10g: JDeveloper Welcome window appears.

Copyright © 1988-2007, Oracle. All rights reserved.

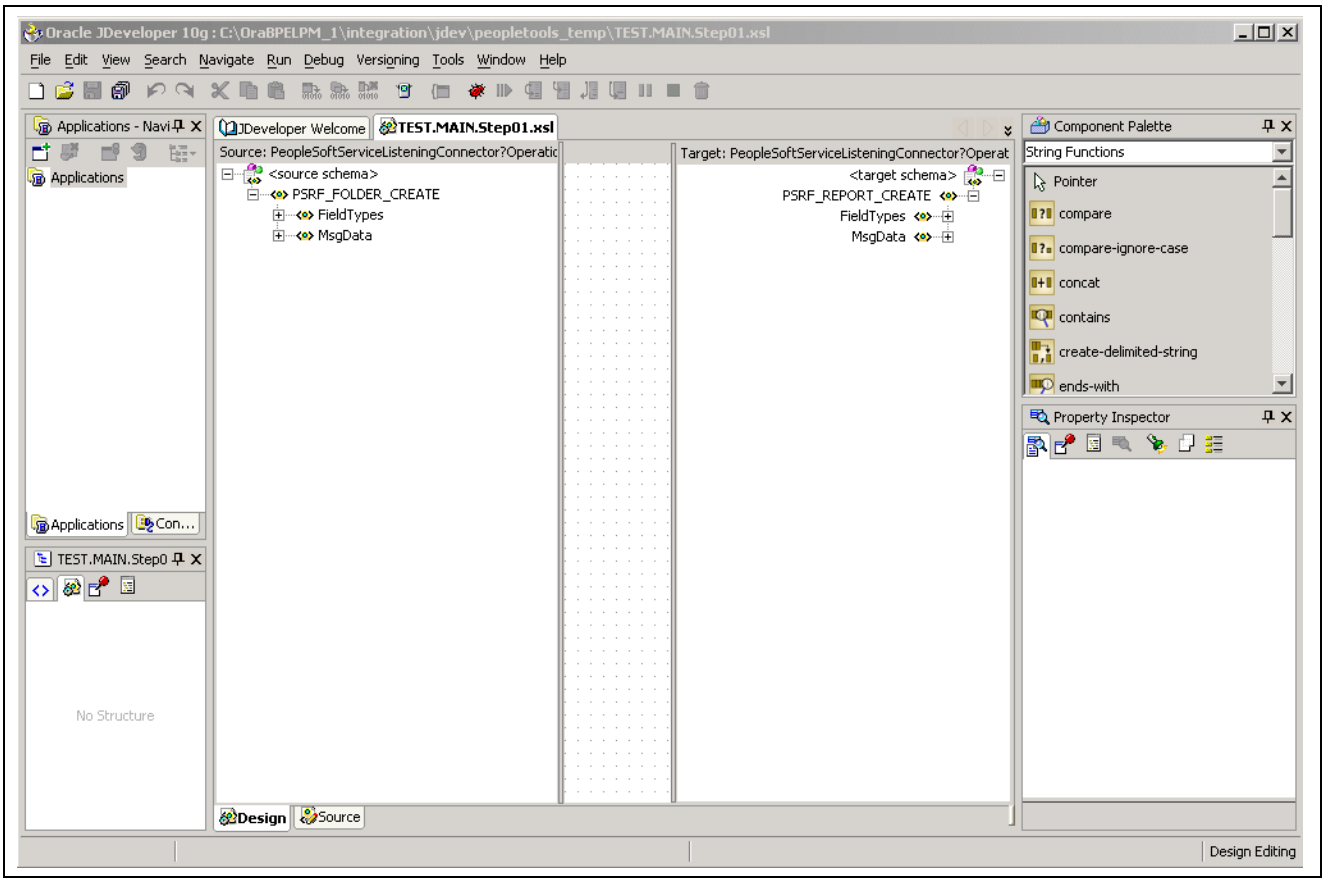


Oracle JDeveloper 10g: JDeveloper Welcome window

Note. The first time you launch Oracle XSL Mapper a Configure Tile Type Associations dialog box appears. While using the mapper you do not work with any Java files and you can disregard the dialog box.

The Oracle JDeveloper 10g: JDeveloper Welcome window displays only the first time you access JDeveloper. When you subsequently open Oracle XSL Mapper, the transform program appears in the Design view.

Copyright © 1988-2007, Oracle. All rights reserved.



Oracle XSL Mapper–Design view

The transform program name in the mapper takes the following format: <transform_program_name>.<section_name>.<step_name>.xsl .

To launch Oracle XSL Mapper:

1. Create a *Transform Only* application engine program and define the program properties described earlier in this chapter.
2. Double-click the XSLT action.
3. Access the Oracle XSL Mapper–Design view.

   When accessing the mapper for the first time, when you double-click the XSLT action and launch Oracle JDeveloper 10g, the Oracle JDeveloper 10g: JDeveloper Welcome window appears. To switch to the Design view, above the Source pane, click the transform file name tab or from the Window menu click the transform file name.

   In subsequent attempts to launch the mapper, double-clicking the XSLT action automatically opens the transform program in the Design view.

Accessing Oracle JDeveloper 10g Documentation and Online Resources

This section provides information for access online Help, documentation and other resources for using Oracle XSL Mapper.

Copyright © 1988-2007, Oracle. All rights reserved.

Online Help

Online Help is available via the Help menu while working with Oracle XSL Mapper.

Additional Oracle JDeveloper 10g Documentation and Online Resources

As mentioned earlier in this section, Oracle XSL Mapper is a plug-in to Oracle JDeveloper 10g. Documentation for the mapper is contained in the Oracle JDeveloper 10g Release 2 (10.1.2) documentation set.

Note. The information provided in this section is current as of the publish date of this PeopleBook.

On the Developer Welcome page of the Oracle XSL Mapper there are links to documentation and online resources located on the Oracle web site.

To access Oracle JDeveloper 10g documentation and resources:

1. Access the Developer Welcome page of the Oracle XSL Mapper.
2. In the Online Resources section, click the JDeveloper's Home Page link.

   The Oracle JDeveloper 10g home page appears.
3. Locate the Technical section located toward the bottom of the page.
4. Click the Documentation link.
5. Under the Oracle JDeveloper 10g Release 2 (10.1.2) Documentation section, select the documentation or resource to view. The options are:
   - *Oracle Application Development Framework Guidelines Manual 10.1.2*.
   - *Oracle Application Development Framework Case Manual 10.1.2*.
   - *Installation Guide 10.1.2*.
   - *Release Notes 10.1.2*.
   - *Release Notes Addendum 10.1.2*.
   - *J2EE Developer Online Help (the main documentation library) 10.1.2* .

Navigating in Oracle XSL Mapper

This discusses how to:

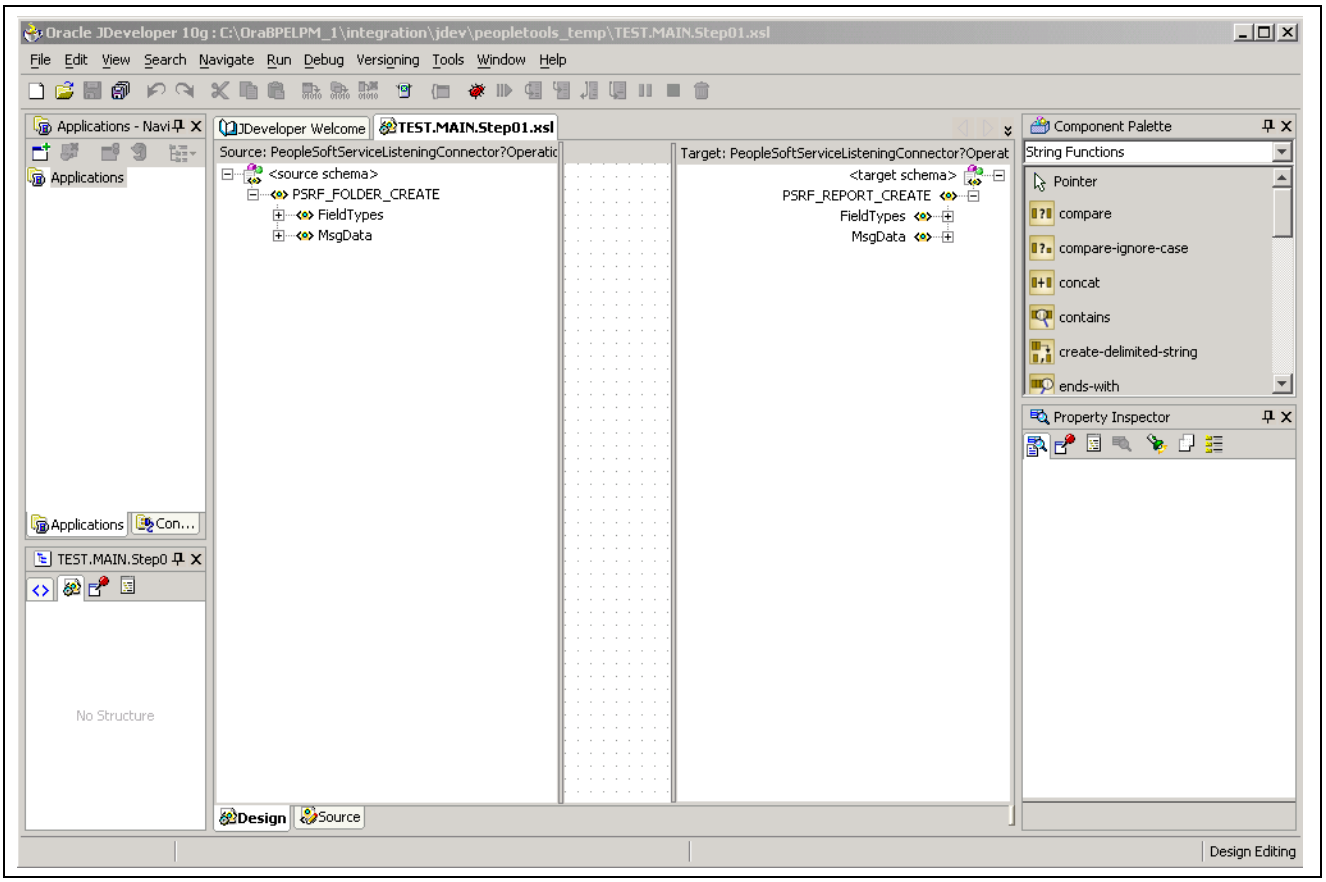
- Navigate in the Design View
- Navigate in the Source View

Note. This section features a brief discussion of some of the key components and areas of the Oracle XSL Mapper development tool. The Oracle JDeveloper 10g documentation provides in-depth details about using the Oracle XSL Mapper.

Navigating in the Design View

To access the Design view of Oracle XSL Mapper, click the Design tab at the bottom of the mapper. The following graphic shows the Oracle XSL Mapper – Design view:

 Copyright © 1988-2007, Oracle. All rights reserved.



Oracle XSL Mapper – Design view

Use the Design view to map records and fields.

A few of the key areas of the Design view are discussed here. For additional information on Design view features, see the Oracle XSL Mapper documentation.

Title Bar

The title bar displays the full path to the Oracle JDeveloper installation location and the name of the current transform program.

Developer Welcome tab

Click the tab to display links to online resources, such as documentation, online demos and code samples.

Transform Program name tab

Click to access the transform program.

Source

The Source pane in the Oracle XSL Mapper main development view provides a hierarchical view of the source or input message and schema.

Click the plus (+) button and the minus (-) button to expand and collapse data shown.

Drag the edges of the pane in or out to adjust the viewing area.

Target

The Target pane in the Oracle XSL Mapper main development view provides a hierarchical view of the target or output message and schema.

Click the plus (+) button and the minus (-) button to expand and collapse data shown.

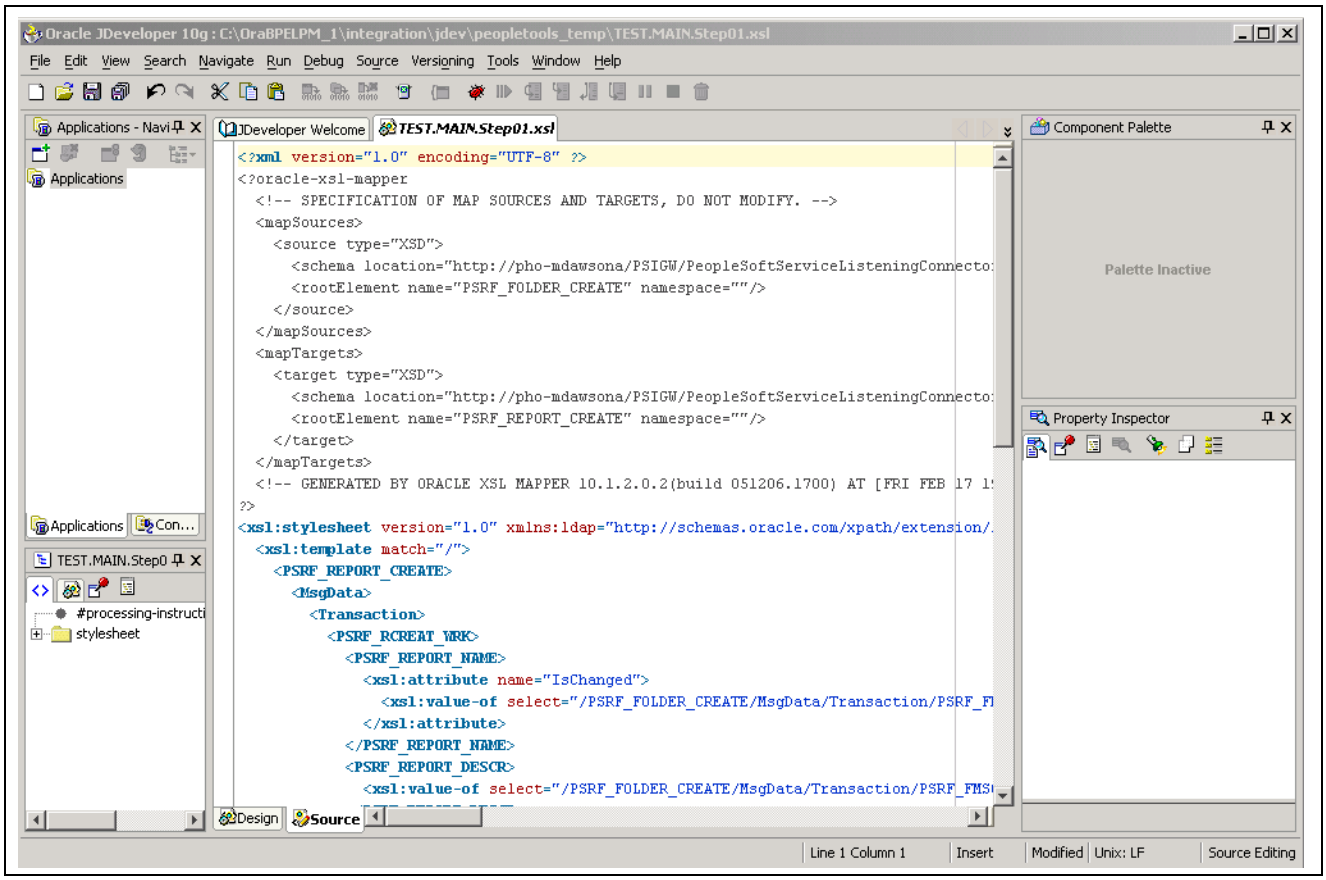
Drag the edges of the pane in or out to adjust the viewing area.

Copyright © 1988-2007, Oracle. All rights reserved.

Design tab	Located at the bottom left-side of the window, click the tab to display the area where you map records and fields.
Source tab	Located at the bottom left-side of the window, click the tab to view and edit the raw XSLT code generated by Oracle XSL Mapper.

Navigating in the Source View

To access the Source view of Oracle XSL Mapper, click the Source tab at the bottom or the mapper. The following graphic shows the Oracle XSL Mapper–Source view:



Oracle XSL Mapper–Source view

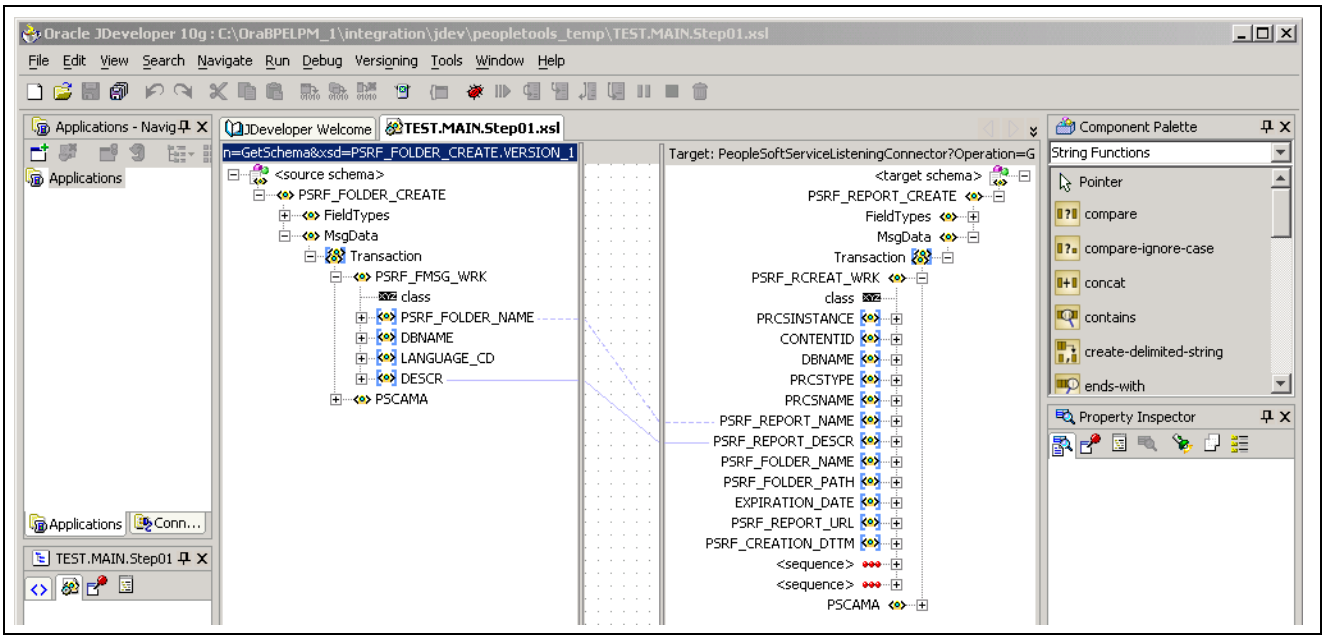
Use the Source view to view and edit the raw XSL code generated by the mapper.

For additional information on Source view features, see the Oracle XSL Mapper documentation.

Mapping Records and Fields

To map records and fields, you drag record names and field names from the source pane to the target pane in the mapper. The following graphic shows and example of mapping several fields:

Copyright © 1988-2007, Oracle. All rights reserved.



Mapping fields in Oracle XSL Mapper–Design view

A solid green line appears as you drag the cursor from the source pane to the target pane. When you release the cursor, the line turns blue to show the association.

To map records and fields:

1. Open the Design view of Oracle XSL Mapper.
2. Expand the contents of both the source and target panes.

   Click the plus (+) button to expand each level or section until all records and fields appear.

   Note. When working with rowset-based messages the content to map is located in the MsgData section.

3. In the source pane, click on the icon to the left of a record or field name and drag it to the name of the record or field in the target pane to which you want to map.

   Repeat this step for each record or field to map.
4. Click the Save button.

Deleting Record and Field Maps

If you have not saved the program and want to delete the last map you made, simply right click in the Design view and select Undo Link. Otherwise, following the instructions in this section.

To delete a record or field map:

1. In the source pane, click the record or field name.
2. From the Edit menu, click Delete.
3. Save the changes.

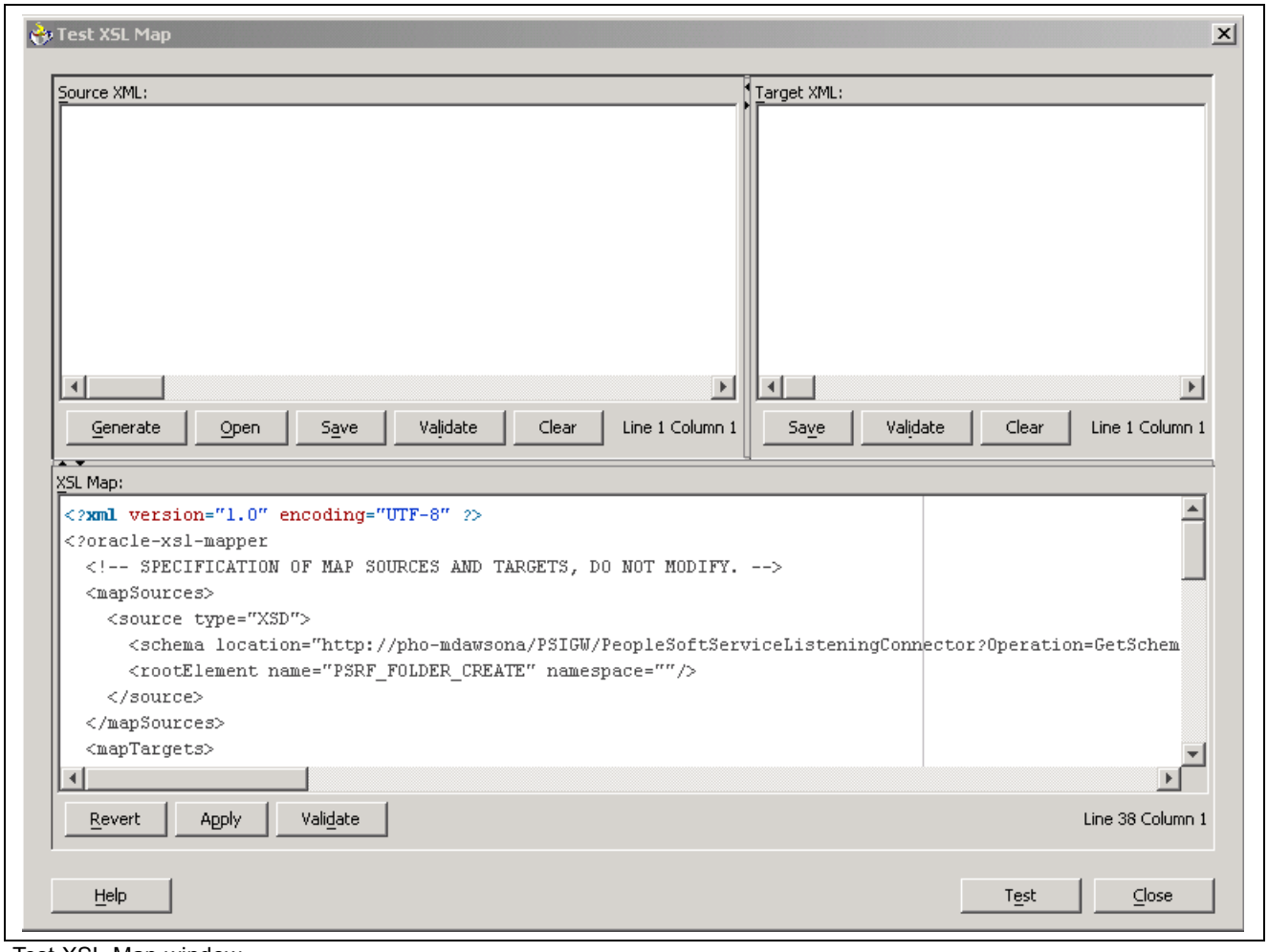
Copyright © 1988-2007, Oracle. All rights reserved.

Viewing Raw XSLT Code

After you have made all of the record and field maps for the program. Click the Source tab at the bottom of the window to generate and view the raw XSL.

Testing XSL Maps

You can test XSL maps for validity that you generate in the mapper using the Test XSL Map window shown in the following graphic:



Test XSL Map window

The code created by the record and field mapping displays in the bottom portion of the window.

Note. This Oracle XSL Mapper test feature tests the validity of the XSLT map code. It does not test the transformation. To test a transformation, use the Transformation Test tool in the PeopleSoft Pure Internet Architecture.

To test XSLT code:

1. Access the Oracle XSL Mapper–Source view.
2. Right-click anywhere in the code area and select Test.

   The Test XSL Map window appears.

Copyright © 1988-2007, Oracle. All rights reserved.

3. In the XSL Map section of the window, click the Validate button.

   A validation success or error message displays.

See Also

Enterprise PeopleTools 8.49 PeopleBook: PeopleSoft Integration Testing Utilities and Tools, "Using the Transformation Test Utility"

Adding and Modifying XSL Map Code

If you add code to XSLT that you create using Oracle XSL Mapper that contains any PeopleSoft-related elements it will not compile successfully, unless you create a configuration file of PeopleSoft elements that the mapper should ignore. After you create this file specify the file location and name in the Oracle XSL Mapper Preferences.

Creating Elements to Ignore Configuration Files

For example, you could add code that contains the following elements:

```
<element name = "psft_function">
<element name = "psft_workingstorage">
```

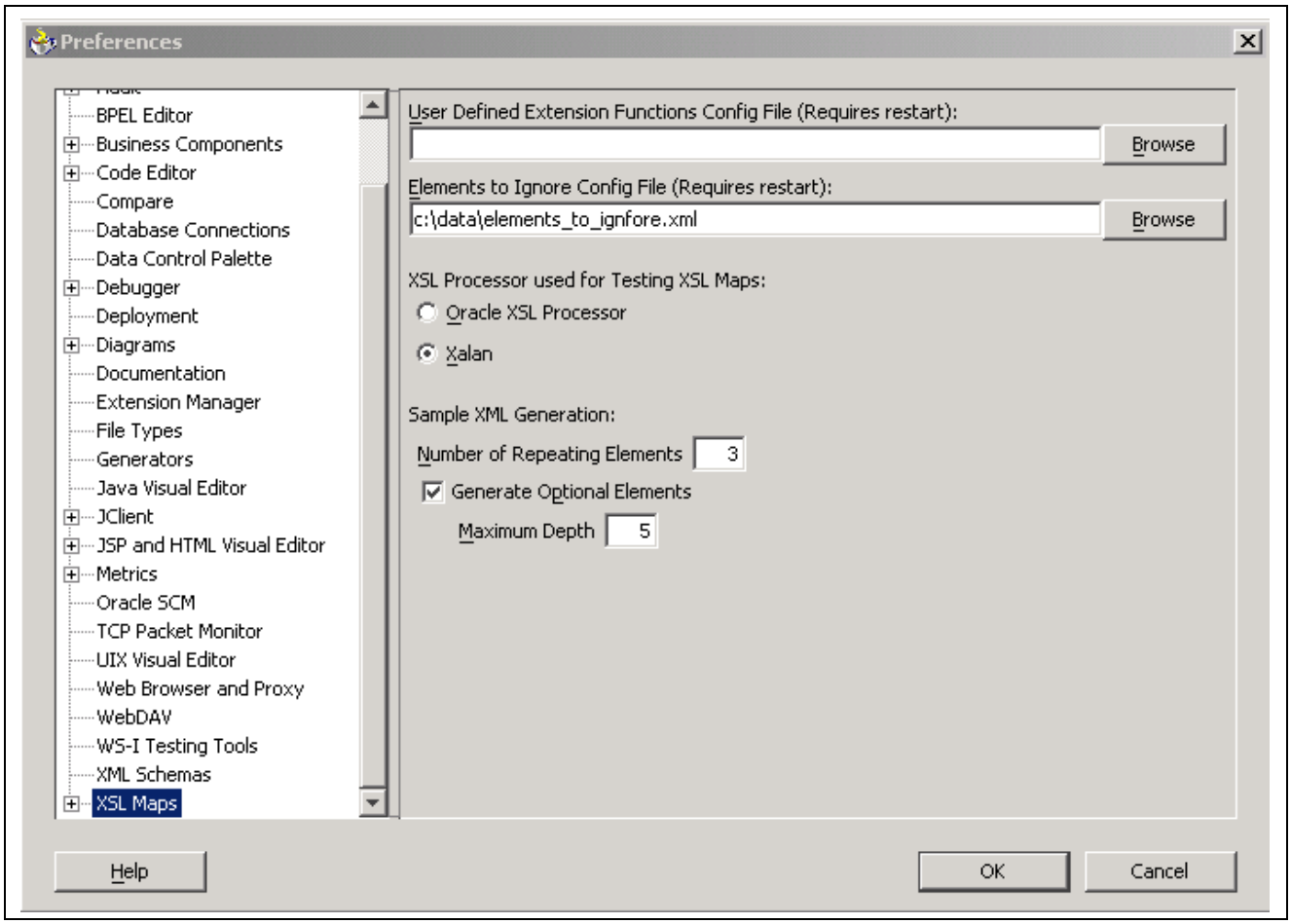
You would create a file using the following format, which tells Oracle XSL Mapper to ignore those elements:

```
<?XML version='1.0' encoding='windows-1252'?>
<elements-to-ignore>
     <element name = "psft_function"/>
     <element name = "psft_workingstorage"/>
</elements-to-ignore>
```

Specifying Elements to Ignore Configuration Files in Oracle XSL Mapper Preferences

Once you create the file of PeopleSoft elements to ignore, enter the file location and name in the mapper Preferences dialog box. The following graphic shows the Preferences dialog box:

Copyright © 1988-2007, Oracle. All rights reserved.



Preferences dialog box

To specify an elements to ignore configuration file in Oracle XSL Mapper preferences:

1. In Oracle XSL Mapper, from the Tools menu, select Preferences.

   The Preferences dialog appears.
2. In the list on the left-side of the window, click XSL Maps.
3. In the Elements to Ignore Conf. File (Requires Restart) field, enter or browse to the location of the elements to ignore configuration file that you created.
4. Click the OK button.

You must restart Oracle JDeveloper 10g for the system to recognize the configuration file.

Filtering Messages

This section provides an overview of message filtering and discusses how to work with a PeopleCode filtering example.

Copyright © 1988-2007, Oracle. All rights reserved.

Understanding Message Filtering

You use filtering to suppress an input message based on its content. For example, you can suppress all inbound purchase order messages that specify order quantities less than the minimum number required for a discount.

Place filtering steps early in your Application Engine transform program; each message suppressed by the filter is one less message for subsequent steps to process.

You must use PeopleCode for filtering, so it'll probably be a distinct step. Because you must use the XmlDoc and XmlNode classes in your PeopleCode transform steps, you can analyze messages in any way that those classes support.

Filtering requires the following actions in your PeopleCode program:

1. Retrieve the message content from the %TransformData system variable.
2. Examine your filtering criteria.
3. If the message meets your criteria, do nothing further. It remains intact in the %TransformData system variable for the next transform program step to process.
4. If the message fails to meet your criteria, replace the entire message content with a single node called *Filter,* containing the reason it failed.

   ```
   <?xml version="1.0"?>
   <Filter>reason_for_failure</Filter>
   ```

5. Set the TransformData *Status* property to *1* to indicate failure.

PeopleSoft Integration Broker examines the *Status* property after each step, and terminates the transform program if its value is *1*. You can then view the message in Integration Broker Monitor and see the reason for the failure.

PeopleCode Filtering Example

The following example of filtering presents an input message, the PeopleCode filtering program, and the resulting output message.

Input Message

This is the input to the filtering step. Notice the line item order quantities (shown emphasized):

```
<?xml version="1.0"?>
<PurchaseOrder>
   <Destination>
      <Address>123 Vine Street</Address>
      <Contact>
         <Name>Joe Smith</Name>
      </Contact>
      <Delivery type="ground">
         <Business>FedEx</Business>
      </Delivery>
   </Destination>
   <Payment>
      <CreditCard cardtype="visa">9999-9999-9999-9999</CreditCard>
   </Payment>
   <LineItems count="2">
```

Copyright © 1988-2007, Oracle. All rights reserved.

```
      <Li locale="en_us" number="1">
         <Quantity>4</Quantity>
         <ProductName>pencil</ProductName>
         <UOM>box</UOM>
      </Li>
      <Li locale="en_us" number="2">
         <Quantity>10</Quantity>
         <ProductName>paper</ProductName>
         <UOM>large box</UOM>
      </Li>
   </LineItems>
</PurchaseOrder>
```

Note. Although this input message isn't in the PeopleSoft rowset-based message format, it is valid XML.

PeopleCode Filtering Program

This filtering program examines the line item order quantities of the input message and generates the output message that follows. The key statements are in bold:

```
/* Get the data from the AE Runtime */
Local TransformData &tempData = %TransformData;
/* Set a temp object to contain the incoming document */
Local XmlDoc &tempDoc = &tempData.XmlDoc;

/* Find the line items quantities contained in the incoming Purchase Order */
Local array of XmlNode &quantities =
 &tempDoc.DocumentElement.FindNodes("LineItems/Li/Quantity");
/* Temp storage of a node */
Local XmlNode &tempNode;

/* Loop through the quantities and make sure they are all above 5 */
For &i = 1 To &quantities.Len
   /* Set the temp node*/
   &tempNode = &quantities [&i];
   /* Make sure the node isn't empty*/
   If ( Not &tempNode.IsNull) Then

      /* Check the value, if not greater than 5 this does not pass filter*/
      If (Value(&tempNode.NodeValue) < 5) Then
         /* Clear out the doc and put in the "Filter" root node */
         If (&tempDoc.ParseXmlString("<?xml version=""1.0""?><Filter/>")) Then
            /* Get the new root node and set the value
            /*  to be the reason for failing filter     */
            &rootNode = &tempDoc.DocumentElement;
            &rootNode.NodeValue = "Line item quantity was found
            that was less than 5!";
            /* Set the status of the transformation to 1 for failed filter*/
            &tempData.Status = 1;
         End-If;
         Break;
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
        End-If

     End-If
End-For;
```

Output Message

This is the result of applying the PeopleCode filtering program:

```
<?xml version="1.0"?>
<Filter>Line item quantity was found that was less than 5!</Filter>
```

Applying Transformations

This section provides an overview of transformation and discusses using XSLT for transformation.

Understanding Transformation

Use a transformation when one node sends a request or response message with a data structure different from the structure required by the other node. One or both of the participating nodes can be PeopleSoft applications. At either end of the transaction, any of the following structure types may be required:

- The PeopleSoft rowset-based message format.
- An XML DOM-compliant non-rowset-based structure. This is generic XML data.
- A SOAP-compliant XML structure. This is also XML DOM-compliant.
- A non-XML structure. Third-party applications are more likely than PeopleSoft applications to require this type.

Your transformation can be between different structure types or between different structures of the same type.

See Also

Chapter 8, “Understanding Supported Message Structures,” page 81

Using XSLT for Transformation

An XSLT transformation simulates the original message structure, then specifies how to treat nodes within that structure. You can:

- Copy the original content of a node without changing anything.
- Define and insert a new version of a node.
- Enter any structure or content directly.
- Eliminate a node by omitting reference to it, or by not inserting anything new in its place.

XSLT Transformation Example

The XSLT wrapper is required.

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

Copyright © 1988-2007, Oracle. All rights reserved.

The primary node tag matches the original message structure by matching its top level content tag, the message name (QE_SYNC_MSG). Between the message template tags, you can insert any structure or content you want. PeopleSoft Integration Broker replaces each *xsl* tag with the data it references, producing a transformed message as the output of the step.

```
<xsl:template match="QE_SYNC_MSG">
    <QE_SYNC_MSG>
       <xsl:copy-of select="FieldTypes"/>
       <MsgData>
          <Transaction>
             <xsl:apply-templates select="MsgData/Transaction/QE_SALES_ORDER"/>
             <xsl:copy-of select="MsgData/Transaction/PSCAMA"/>
          </Transaction>
       </MsgData>
    </QE_SYNC_MSG>
 </xsl:template>
```

The following node example is defined to match a record in the input message by its top level content tag, the record name (QE_SALES_ORDER). This template is applied by the *xsl:apply-templates* tag of the preceding node (shown emphasized).

Between the record template tags, you can insert any structure or content you want. In this example, *90* is prepended to the QE_ACCT_ID value, and the QE_ACCOUNT_NAME field is renamed to QE_ACCOUNT (shown emphasized). Also, any existing value in the DESCRLONG field is removed, and the remaining fields are passed through with their original values.

```
<xsl:template match="QE_SALES_ORDER">
    <QE_SALES_ORDER><xsl:attribute name="class">
    <xsl:value-of select="@class"/></xsl:attribute>
       <xsl:variable name="temp" select="QE_ACCT_ID"/>
       <QE_ACCT_ID><xsl:value-of select="concat(90,$temp)"/></QE_ACCT_ID>
       <QE_ACCOUNT><xsl:value-of select="QE_ACCOUNT_NAME"/></QE_ACCOUNT>
       <QE_ADDRESS><xsl:value-of select="QE_ADDRESS"/></QE_ADDRESS>
       <QE_PHONE><xsl:value-of select="QE_PHONE"/></QE_PHONE>
       <QE_FROMROWSET/>
       <QE_TOROWSET/>
       <QE_SEND_SOA_BTN/>
       <QE_SEND_SOS_BTN/>
       <DESCRLONG></DESCRLONG>
    </QE_SALES_ORDER>
 </xsl:template>
```

Finally, you need the closing wrapper tag:

```
</xsl:stylesheet>
```

Note. You can find more information about XSLT at the World Wide Web Consortium (W3C) web site.

Note. A working transformation example using XSLT is provided in the PeopleTools SDK. The location of the example is <PS_HOME> \sdk\pstransform\samples\TRANSFORMTST.xml.

See Also

http://www.w3.org/Style/XSL/

Copyright © 1988-2007, Oracle. All rights reserved.

Performing Data Translation

This section provides an overview of data translation and discusses how to:

- Define codeset groups.
- Define codesets.
- Define codeset values.
- Import and export codesets between databases.
- Delete codesets.
- Use XSLT for data translation.
- Work with an XSLT translation example.
- Work with a PeopleCode translation example.

Understanding Data Translation

Use data translation to modify message content rather than structure, although you can also make local structural changes. It's most appropriate when the sending and receiving systems use different field values, or different combinations of fields and their values, to represent the same information.

Following is a sample scenario:

- Application A transmits customer names in four fields: *Title, First, Middle,* and *Last*.
- Application B uses two fields: *Last* and *First.* It doesn't use a title, but includes the middle name as part of the *First* field.
- Application C uses only one field: *AccountID.*

Clearly the representation used by one application won't be understood by either of the other two. PeopleSoft Integration Broker can apply a transform program to translate each of these representations into a version appropriate to the receiving application.

One Integration Broker node can store in its codeset repository the equivalent fields and values used by another node. When it receives a message from the other node containing a customer name, it can use its codeset repository to translate the information into the form it prefers. It can likewise reverse the process for messages outbound to the other node.

For a given integration, you can allocate the responsibility for performing data translation in different ways. You can distribute the translation activity among the participating nodes, or you can designate one Integration Broker node to do all the data translation as a hub, whether the messages are inbound, outbound, or being redirected between the other nodes. Using a single node, if possible, can reduce the need for duplicating repository data.

Data Translation Elements

The following elements constitute the codeset repository, managed as PeopleSoft Pure Internet Architecture components:

Codeset group Maintains a list of the significant data fields and their values that a particular node might send in an initial message. These are name/value pairs a translation program might find (match) and use as the basis for determining what the

Copyright © 1988-2007, Oracle. All rights reserved.

result message should contain. These name/value pairs are known as *match names* and *match values.*

Each PeopleSoft Integration Broker node that requires data translation must belong to a codeset group.

Codeset A specific set of match name/match value pairs selected from an existing codeset group. The selected name/value pairs are the basis for possible field value combinations that you want to match in a message, and to which your translation program can respond by modifying the message content. Each codeset typically represents one set of fields that require translation for a given message.

Codeset values A codeset value is a named value you define, also known as a *return value.* Your translation program can output the return value as a result of matching a specific combination of match values from a codeset.

You associate multiple combinations of codeset values with the combination of an initial codeset group, a codeset from that initial group, and a result codeset group. For each permutation of match values selected from the codeset, you define a different combination of codeset values to apply to your result message.

The other key element of data translation is your translation program, which invokes the codesets and codeset values you've defined.

Data Translation Development Sequence

You must initially define these elements in a particular order:

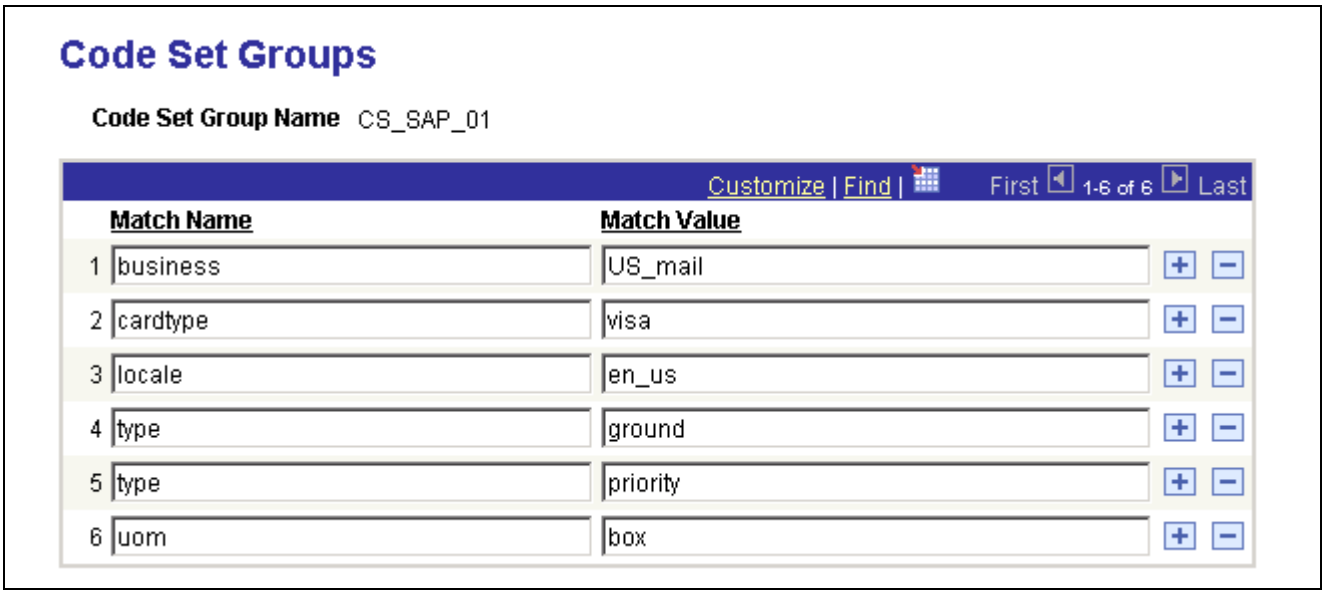
1. Two codeset groups.
2. A codeset based on one of the codeset groups.
3. A set of codeset values.
4. A data translation program, in XSLT or PeopleCode.

However, it's unlikely that you'll be able to fully define any of these elements without some trial and error. You may find you'll have to modify and extend each element in turn as you develop your data translation program.

Defining Codeset Groups

Use the Codeset Groups page in the Codeset Groups component (IB_CODESETGROUP) to define codeset groups. To access the Codeset Groups page, select PeopleTools, Integration Broker, Integration Setup, Codesets, Codeset Groups.

 Copyright © 1988-2007, Oracle. All rights reserved.



Codeset Group page

To define a codeset group:

1. Add a new value, enter a codeset group name, and click Add.

   Enter a name that reflects a common quality of the nodes you plan to assign to this group; for example, the name of the software they all use to manage shipping.

   The Codeset Group page appears.

2. Add a new row.
3. Enter a match name.

   This is the name of a data field that might be part of a message sent by a node belonging to this codeset group. You don't have to create an entry for every field, just the ones that you'll need to translate or use for reference in a translation.

4. Enter a match value.

   This is one of the possible values of the data field represented by the match name.

5. Repeat steps 2 through 4 for each significant name/value pair that you expect to appear in a message.

   This doesn't need to be all possible values of all of the message fields, just the names and values you expect to require translation.

6. Assign one or more nodes to this codeset group.

   Every initial and result node involved in a data translation must belong to a codeset group. You must assign each participating node to an appropriate codeset group by an entry in its node properties.

Note. The assignment for each node is required only in the database of the node performing the data translation. This translating node needn't be either the source or the target. Multiple nodes that represent data the same way may be assigned to the same codeset group.

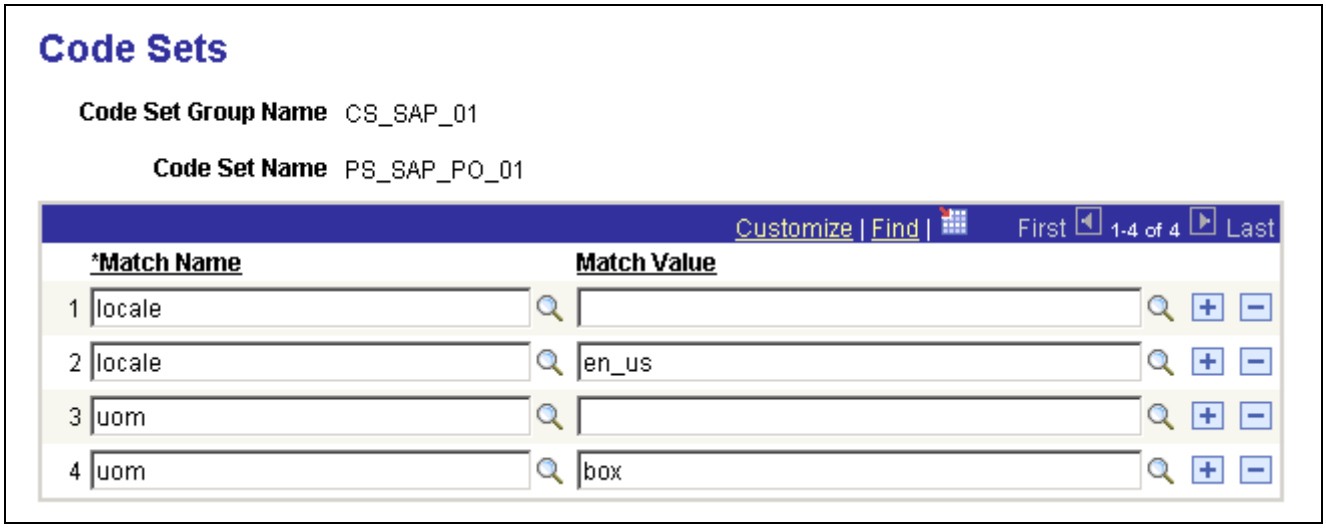
See Also

Chapter 19, "Adding and Configuring Nodes," Configuring Nodes, page 363

Copyright © 1988-2007, Oracle. All rights reserved.

Defining Codesets

Use the Codesets page in the Codesets component (IB_CODESET). Select PeopleTools, Integration Broker, Integration Setup, Codesets, Codesets to access the Codeset page.



Codeset page

To define a codeset:

1. Add a new value and enter a codeset group name on which to base this codeset.
2. Enter a codeset name and click Add.

   Enter a name that reflects the purpose of this codeset; for example, *SAP_SHIP_METHOD*, to translate the representation of a shipping method in a message. The Codesets page appears.
3. Add a new row.
4. Select a match name from the set defined for the associated codeset group.
5. Select a match value from the set defined for the selected match name.

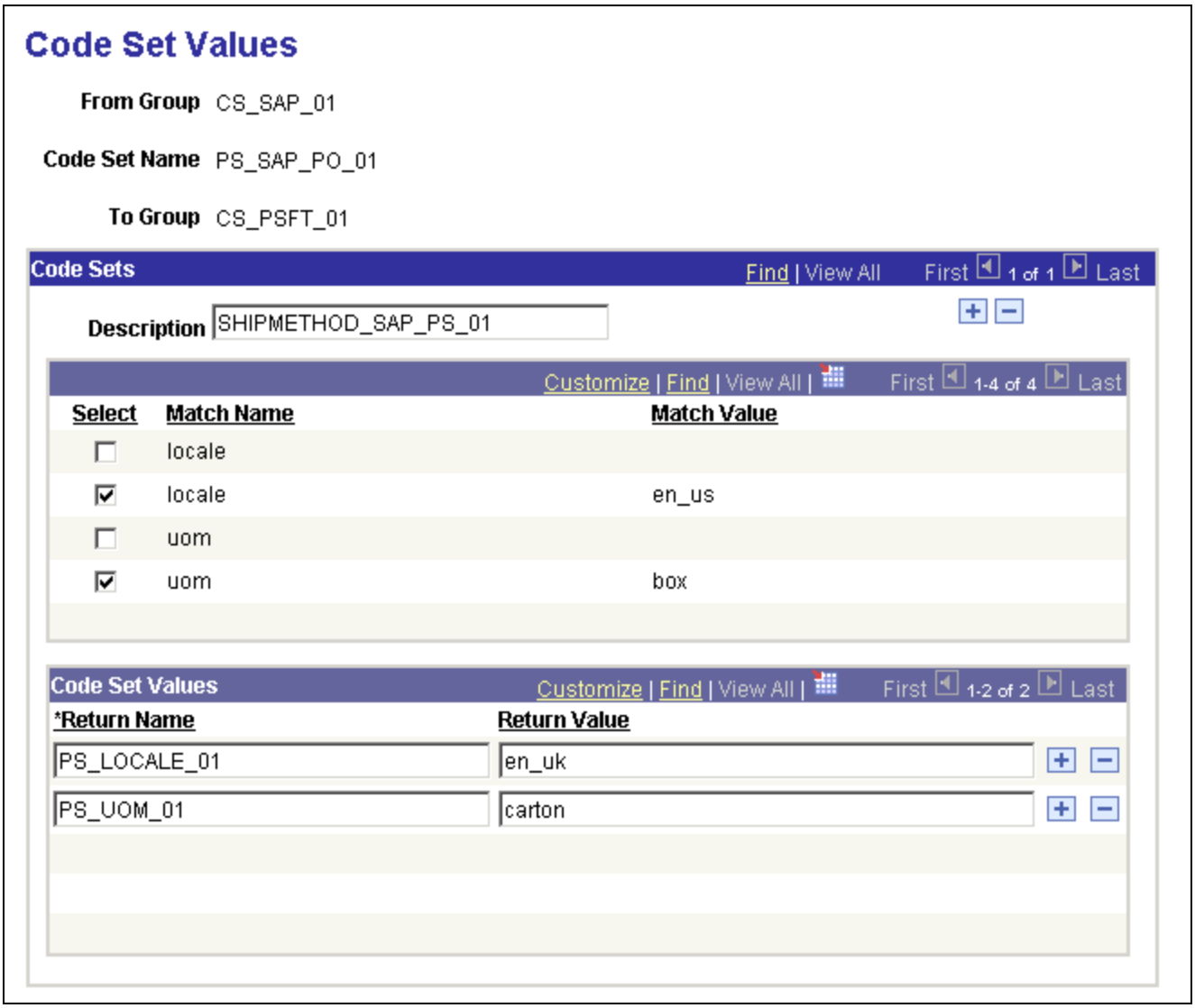
   Note. You can leave the value blank. If so, you should do the same for each match name in this codeset, in addition to any other values you select for them. A combination consisting of *all* blank values is treated as a wild card by PeopleSoft Integration Broker, which enables it to respond to unanticipated values specified in your translation program with default behavior that you define.
6. Repeat steps 3 through 5 to enter all the name/value pairs that may need to be matched.

   The name/value pairs you select should encompass only the possible value combinations that your translation program needs to match for a single translation. You define a different codeset for each translation based on this codeset group.

Defining Codeset Values

Use the Codeset Values page in the Codeset Values component (IB_CODESETVAL). Select PeopleTools, Integration Broker, Integration Setup, Codesets, Codeset Values to access the Codeset Values page.

Copyright © 1988-2007, Oracle. All rights reserved.



Codeset Values page

To define codeset values:

1. Add a new value and select a codeset group name for the *From* group.

   This is the codeset group to which the initial node belongs.

2. Select a codeset name from the codesets based on the group you selected.

   This is the codeset whose match name/match value permutations you wish to match.

3. Select a codeset group name for the *To* group.

   This is the codeset group to which the result node belongs.

4. Click Add.

   The Codeset Values page appears. The upper grid contains the selected codeset's match name/match value pairs, and the lower grid contains the return values you specify. Each permutation that you define has its own Description field, which can help you distinguish between permutations that may be subtly different from each other.

Copyright © 1988-2007, Oracle. All rights reserved.

Note. To configure an existing codeset values definition, enter its *From* group, codeset name and *To* group on the search page.

5. Select check boxes to define a permutation of match name/match value pairs.

   For each match name, you can select at most one match value.

   A permutation consisting of all blank values serves as a wild card; it matches any input value combination that isn't matched by any other permutation. However, a permutation with some blank and some non-blank values works differently; it requires the names with blank values to actually match blank field values in the input data.

   Note. You'll generally define only permutations that you expect the input data to contain, but make sure you allow for unforeseen match values by including permutations with all blank values. You can then specify default return values for those permutations. With a large number of match names in the codeset, you can make sure to catch all unforeseen combinations by defining a permutation with all blank match values.

6. In the Code Set Values grid, enter a return name, and a return value for that name.

   You can use any return name you want, because only your codeset translation program refers to it. Your translation program can use the return value as a field value or as a node name in the output data.

   Important! The set of return names you define *must be identical* for all of the permutations of match name/match value pairs for the current codeset in this definition. Your translation program invokes the codeset and applies the return names from this definition, but it can't anticipate which permutations will be matched, or which actual return values it's applying — just the return names.

7. (Optional.) In the Code Set Values grid, add a new row and repeat step 6.

   Add as many return name/return value pairs as you need for your output based on the current permutation. If the permutation is matched in the input data, the code set values you define for that permutation become available for you to call and insert in the output data.

8. (Optional.) At the top level of this page, add a new row and repeat steps 5 through 7.

   This inserts a new permutation row, in which you can define a different permutation of match name/match value pairs that you expect for the current codeset. For each permutation, you'll define a separate, independent set of codeset values.

Importing and Exporting Codesets Between Databases

PeopleSoft provides two Data Mover scripts that you can use to import and export codesets between databases:

- CODESET_DELETE_IMPORT.DMS

  Use this script to purge and then import codeset data into a target database.

- CODESET_EXPORT.DMS

  Use this script to export codeset data from a source database to a target database.

Deleting Codesets

Before you delete a codeset, you must delete any codeset values associated with it.

 Copyright © 1988-2007, Oracle. All rights reserved.

Deleting Codeset Values

To delete codeset values for a codeset:

1. Select PeopleTools, Integration Broker, Integration Setup, Codesets, Codeset Values. The Codeset Values page displays.
2. In the Find an Existing Value tab, in the Codeset Name field, enter the name of the code set you want to delete, or use the Lookup button to locate it.
3. In the Codeset Values section, clear the Select box for each match name corresponding to the code set match name you want to delete, or click the minus (-) button to delete the entire code set scroll area.

   Repeat this process for as many codeset match names that are used.
4. Click the Save button.

Deleting Codesets

To delete a codeset:

1. Select PeopleTools, Integration Broker, Integration Setup, Codesets, Codeset.
2. Select the codeset to delete. The Codeset page displays.
3. Locate the row that contains the codeset you want to delete, and click the minus (-) button on that row.
4. Click the Save button.

Using XSLT for Data Translation

Once you've defined the match name/match value permutations for a given codeset and defined the return values for those permutations, you can write an XSLT translation program that invokes the codeset and applies the return values.

An XSLT translation is based on XSLT transformation structure. However, although you could combine both tasks into a single program, it's better to keep them separate for easier understanding and maintenance.

Psft_function Nodes

To implement data translation capability, PeopleSoft Integration Broker provides a custom XSLT tag called *psft_function.* Each *psft_function* node in your program comprises a single instance of data translation that invokes a particular codeset and applies a specified set of codeset values.

Note. You can insert a *psft_function* node anywhere inside the XSLT template containing the fields you want to translate. However, you'll find it easiest to place it at or near the point in the template where the return values will go, to avoid having to specify a complex path to that location.

The *psft_function* tag has the following attributes:

Attribute	Use
name	Set to *codeset.*

Copyright © 1988-2007, Oracle. All rights reserved.

Attribute	Use
codesetname	Identifies the codeset whose name/value permutations you want to match in the input data. The routing that invokes this transform program identifies the initial and result nodes involved, and PeopleSoft Integration Broker examines their definitions to determine the *From* group and *To* group. The combination of these two keys and the codeset name identifies the codeset values definition to apply.
source	(Optional.) Overrides the name of the initial node specified by the routing. PeopleSoft Integration Broker uses the specified node's codeset group as the *From* group key, thus invoking a different codeset values definition.
dest	(Optional.) Overrides the name of the result node specified by the routing. PeopleSoft Integration Broker uses the specified node's codeset group as the *To* group key, thus invoking a different codeset values definition.

Note. The *source* and *dest* attributes don't change the initial and result nodes specified in the routing; they just invoke the codeset groups to which those nodes belong.

Following is an example of *psft_function* using all of its attributes:

```
<psft_function name="codeset" codesetname="PS_SAP_PO_01"
 source="SAP_02" dest="PSFT_03">...</psft_function>
```

Parm and Value Nodes

The *psft_function* node can contain two tags, *parm* and *value:*

- Use the *parm* tag to specify a match name from the codeset values definition that you specified for this translation. You do this with the tag's only attribute: *name.* Set this to a match name from the codeset values definition.

  The *parm* node should contain a match value, usually specified as an *xsl:value-of* tag that identifies where the value resides in the input data. Use one *parm* node for each distinct match name in the codeset values definition.

- Use the *value* tag to specify a return name from the codeset values definition that you specified for this translation. Also use the *value* tag to identify where to place the return value assigned to that return name for the matched permutation and how to apply that value.

  Use one *value* node for each return name in the codeset values definition that you want in your output.

Value Tag Attributes

The *value* tag has the following attributes:

 Copyright © 1988-2007, Oracle. All rights reserved.

Attribute	Use
name	Identifies a return name from the codeset values definition you specified for this translation. The return value assigned to this return name can be used as a data value or as a node name in your output depending on the attributes you specify.
select	Identifies an XSLT path (XPATH) to the location where the return value should be applied in the output data.
createIfDNE	(Optional.) Set to *yes* to ensure that the node specified by the *select* attribute is created if it does not exist yet. The return value is inserted as the value of that node.
createNodeFromValue	(Optional.) Set to *yes* to use the return value as the name of a new node, created where the *select* attribute specifies. The *value* tag can contain a valid XSLT value for that node, usually specified as an *xsl:value-of* tag that identifies where the value resides in the input data.

Following is an example of a *value* node:

```
<value name="PS_RET_01" select="." createNodeFromValue="yes"><xsl:value-of
 select="CreditCard"/></value>
```

See Also

Chapter 20, “Applying Filtering, Transformation and Translation,” Using XSLT for Transformation, page 397

http://www.w3.org/Style/XSL/

XSLT Translation Example

The following example of XSLT data translation presents an example input message, the XSLT translation program, and the resulting output message.

Input Message

This is the input to the XSLT translation:

```
<?xml version="1.0"?>
<PurchaseOrder>
   <Destination>
      <Address>123 Vine Street</Address>
      <Contact>
         <Name>Joe Smith</Name>
      </Contact>
      <Delivery type="ground">
         <Business>FedEx</Business>
      </Delivery>
   </Destination>
   <Payment>
```

Copyright © 1988-2007, Oracle. All rights reserved.

```
      <CreditCard cardtype="visa">9999-9999-9999-9999</CreditCard>
   </Payment>
   <LineItems count="2">
      <Li locale="en_us" number="1">
         <Quantity>15</Quantity>
         <ProductName>pencil</ProductName>
         <UOM>box</UOM>
      </Li>
      <Li locale="en_us" number="2">
         <Quantity>10</Quantity>
         <ProductName>paper</ProductName>
         <UOM>large box</UOM>
      </Li>
   </LineItems>
</PurchaseOrder>
```

Note. Although this input message isn't in the PeopleSoft rowset-based message format, it is valid XML.

XSLT Data Translation Program

This translation program processes the input message in this example and generates the output message that follows. The statements shown emphasized demonstrate some uses of the *psft_function* node:

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
   <xsl:template match="PurchaseOrder">
      <po>
         <xsl:apply-templates/>
      </po>
   </xsl:template>

   <xsl:template match="Destination">
      <dest>
         <address><xsl:value-of select="Address"/></address>
         <name><xsl:value-of select="Contact/Name"/></name>
         <delivery>
            <type>
               <psft_function name="codeset" codesetname="PS_SAP_PO_03"
               dest="PSFT_03">
                  <parm name="type"><xsl:value-of select="Delivery/@type"/>
                  </parm>
                  <value name="PS_RET_01" select="."/>
               </psft_function>
            </type>
            <carrier>
               <psft_function name="codeset" codesetname="PS_SAP_PO_03"
               source="SAP_03">
                  <parm name="Business"><xsl:value-of select="Delivery/
                  Business"/></parm>
                  <value name="PS_RET_01" select="."/>
               </psft_function>
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
            </carrier>
         </delivery>
      </dest>
   </xsl:template>

   <xsl:template match="Payment">
      <payment>
         <psft_function name="codeset" codesetname="PS_SAP_PO_02">
             <parm name="cardtype"><xsl:value-of select="CreditCard/
             @cardtype"/></parm>
             <value name="PS_RET_01" select="."
             createNodeFromValue="yes"><xsl:value-of select="CreditCard"/>
             </value>
         </psft_function>
      </payment>
   </xsl:template>

   <xsl:template match="Li">
      <li><xsl:attribute name="id"><xsl:value-of select="@number"/></xsl:attribute>
         <name><xsl:value-of select="ProductName"/></name>
         <qty><xsl:value-of select="Quantity"/></qty>
         <uom>
            <psft_function name="codeset" codesetname="PS_SAP_PO_01">
                <parm name="locale"><xsl:value-of select="@locale"/></parm>
                <parm name="uom"><xsl:value-of select="UOM"/></parm>
                <value name="PS_RET_01" select="."/>
                <value name="PS_RET_02" select="../type" createIfDNE="yes"/>
            </psft_function>
         </uom>
      </li>
   </xsl:template>
</xsl:stylesheet>
```

Output Message

This is the result of applying the XSLT translation:

```
<po>
   <li>
      <name>pencil</name>
      <qty>15</qty>
      <uom>Carton</uom>
      <type>Bic</type>
   </li>
   <li>
      <name>paper</name>
      <qty>10</qty>
      <uom>Box</uom>
      <type>Bic</type>
   </li>
   <dest>
```

Copyright © 1988-2007, Oracle. All rights reserved.

```
      <address>123 Vine Street</address>
      <name>Joe Smith</name>
      <delivery>
         <type>Ground</type>
         <carrier>Federal Express</carrier>
      </delivery>
   </dest>
   <payment>
      <VISA>4024-9920-9892-8982</VISA>
   </payment>
</po>
```

PeopleCode Translation Example

Although XSLT is the recommended language for using the codeset repository to translate message data, you can use PeopleCode for this purpose as well. Because XSLT works only with XML DOM-compliant message data, you must use PeopleCode if the message you're translating contains non-XML data, including formats like comma separated values (CSV).

Once you've defined the match name/match value permutations for a codeset with respect to a given target codeset group and defined the return values for those permutations, you can write a PeopleCode translation program that invokes that codeset and applies the return values.

FindCodeSetValues Built-in Function

To implement data translation capability, Integration Broker provides a PeopleCode built-in function called *FindCodeSetValues*, which takes four parameters and returns a two dimensional array.

The following example of PeopleCode data translation presents an example input message, the PeopleCode translation program, and the resulting output message.

Input Message

This is the input to the PeopleCode translation:

```
<?xml version="1.0"?>
<Header>
  <LANGUAGE_CODE>en_us</LANGUAGE_CODE>
  <STATUS_CODE>0</STATUS_CODE>
</Header>
```

PeopleCode Data Translation Program

This translation program processes the input message in this example, and generates the output message that follows. The statement shown emphasized demonstrates the use of the FindCodeSetValues function:

```
/* Get the data from the AE Runtime */
Local TransformData &incomingData = %TransformData;

/* Set a temp object to contain the incoming document */
Local XmlDoc &tempDoc = &incomingData.XmlDoc;

/* Find the Language and status codes value*/
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
Local string &langCode = &tempDoc.DocumentElement.FindNode("LANGUAGE_CODE").
   Node Value;
Local string &statusCode = &tempDoc.DocumentElement.FindNode("STATUS_CODE").
   Node Value;

/* Create an array to hold the name value pairs */
Local array of array of string &inNameValuePairsAry;

/* Load the array with some values */
&inNameValuePairsAry = CreateArray(CreateArray("LANG", &langCode),
 CreateArray("STATUS", &statusCode));

/* Find the codeset values */
&outAry = FindCodeSetValues("STATUS_CHANGE", &inNameValuePairsAry,
 &incomingData.SourceNode, &incomingData.DestNode);

/* Create the new output doc */
If &tempDoc.ParseXmlString("<?xml version=""1.0""?><NewHeader/>") Then

   /* Make sure something was returned */
   If &outAry.Len > 0 Then

      /* Create the new Status Code Node */
      Local XmlNode &statusNode = &tempDoc.DocumentElement.AddElement("STATUS");

      /* Since this is a 2D array, get the Return Value*/
      &statusNode.NodeValue = &outAry [1][2];
   End-If;
End-If;
```

Output Message

This is the result of applying the PeopleCode translation:

```
<?xml version="1.0"?>
<NewHeader>
   <STATUS>Open</STATUS>
</NewHeader>
```

See Also

Enterprise PeopleTools 8.49 PeopleBook: PeopleCode Language Reference, "PeopleCode Built-in Functions," FindCodeSetValues

Copyright © 1988-2007, Oracle. All rights reserved.

Terminating Transformation Programs

If you need to terminate a transform program for reasons that aren't considered error conditions by PeopleSoft Integration Broker, you can use a PeopleCode step to force the transform program to terminate and generate a readable error message as well.

To generate an error:

1. Replace the entire message content with a single node called *Error*, containing the reason for the error.

```
<?xml version="1.0"?>
<Error>reason_for_error</Error>
```

2. Set the TransformData *Status* property to *2* to indicate error status.

PeopleSoft Integration Broker examines the *Status* property after each step and terminates the transform program if its value is *2*. You can then view the message in Service Operations Monitor and see the reason for the error.

Note. If an XSLT or PeopleCode step fails for reasons that you haven't taken into account, Integration Broker automatically sets the *Status* property to *2* and aborts the transform program, but you can't provide your own error message.

Copyright © 1988-2007, Oracle. All rights reserved.

CHAPTER 21

Using the Service Operations Monitor

This chapter discusses how to:

- Filter asynchronous and synchronous service operations data.
- Monitor asynchronous service operations.
- View asynchronous service operations details.
- Monitor synchronous service operations.
- View synchronous service operation details.
- Resubmit and cancel service operations for processing.
- View service operation IB info data.
- View service operation errors.
- View and edit service operation XML.
- View service operation nonrepudiation signature information.
- Run batch error notification processes.
- Archive service operation instances.
- Run batch service operations archiving processes.
- View system performance statistics.
- Manage pub/sub server domains.
- Set up domain failover.
- Manage down nodes.
- Pause, test and ping nodes.
- Pause and start queues.
- Clean up orphaned data from segment batch processing errors.
- Used custom-defined components to view service operations data.
- Purge runtime monitor tables.
- Use the Service Operations Monitor component interface.
- Use PeopleCode to read and write errors to the asynchronous messaging error queue.

Understanding the Service Operations Monitor

This section provides and overview of the Service Operations Monitor and discusses:

Copyright © 1988-2007, Oracle. All rights reserved.

- Service Operations Monitor security.
- Service Operations Monitor features and components.

Service Operations Monitor Security

Upon accessing the monitor, you can see a list of all transactions in the system, but to see specific information about a transaction and to view transaction details, you must have permission to the service operation.

Service Operations Monitor Features and Components

System administrators use the Service Operations Monitor to monitor asynchronous and synchronous service operations information, node status, queue status, manage domains and more.

Features

The Service Operations Monitor provides:

- Status on queues, nodes, and individual service operations.

  You can also view and edit service operation XML.
- Control and administration of domains that have publication and subscription (pub/sub) servers running against the current database.

  You can activate or deactivate domains, recover from stalls, and so forth.
- Workflow notification of error messages and archival of service operations.
- Batch processes for error notification and service operation archival.

Components

There are thirteen components associated with the Service Operations Monitor that are located within Monitor and Administration menus in the PeopleSoft Pure Internet Architecture navigation structure.

The following components are located under the Monitor menu. Access them by selecting PeopleTools, Integration Broker, Service Operations Monitor, Monitoring.

Asynchronous Services	Use this component to monitor asynchronous service operations and view information about service operation instances, publication contracts and subscription contracts.
Asynchronous Details	View asynchronous service operation details, including information about the service operation instance, its publication or subscription contracts, error messages, and service operation instance XML. If transformations have been applied to the service operation, you can view the transformed XML for the publication and subscription contracts.
Synchronous Services	Use this component to view synchronous service operations.
Synchronous Details	View synchronous service operation details and service operation errors, and view request and response XML (before or after transformation).
Error Notification	Run batch processes to receive notification of issues affecting the messaging system.
Archive Monitor Data	Run the batch process to archive service operations.

Copyright © 1988-2007, Oracle. All rights reserved.

Statistics	View runtime performance statistics for asynchronous and synchronous transactions that flow through the messaging system. View statistics in numeric or graphical format.

The following components are located under the Administration menu in the PeopleSoft Pure Internet Architecture navigation structure. Access them by selecting PeopleTools, Integration Broker, Service Operations Monitor, Administration.

Domain Status	View and maintain domain status and activate pub/sub server domain. Use this component to also setup domain failover.
Node Status	View node status. Ping node.
Queue Status	View and maintain queue status.
Segment Cleanup	Delete orphaned data after segment batch processing errors.
User Details Component	Define a custom component to review service operation transaction details for a specific service operation.
Monitor Setup Options	Define parameters for using the system performance statistics feature and for setting the data length view limit for loading XML data into the monitor.

Filtering Asynchronous and Synchronous Service Operations Data

This section discusses:

- Selecting filtering criteria.
- Saving filtering criteria.

Understanding Filtering Asynchronous and Synchronous Service Operations Data

Before you begin monitoring the integration system, there are a few general guidelines that enable you to quickly drill down to the information you need.

When monitoring asynchronous and synchronous service operations, the Service Operations Monitor provides information about the entire integration system, you need to understand how to filter the information to reduce the number of items. For instance, rather than sifting through every service operation in the entire system, the Service Operations Monitor enables you to filter by publishing node, queue, service operation name, publish date and time, live and archived service operations, and so on.

Selecting Filtering Criteria

When you filter data in the Asynchronous Services component or the Synchronous Services component, the value you set on one page in the component is carried forward to other pages in the component.

Copyright © 1988-2007, Oracle. All rights reserved.

See Also

Chapter 21, "Using the Service Operations Monitor," Filtering Asynchronous Service Operations Data, page 419

Chapter 21, "Using the Service Operations Monitor," Filtering Synchronous Service Operations Data, page 436

Saving Filtering Selections

You can save your filtering options so that the next time you use it, your previous filtering choices are set automatically.

To save filtering selections:

1. Select the filtering options on one of the Asynchronous Services or Synchronous Services component pages.
2. Click Refresh button.

   Clicking Refresh not only refreshes the page according to the most recent filtering selections, it also saves the most recent filtering selections to the database. The system then associates a given set of filtering selections with your user ID. The next time that you sign in and launch the Services Operation Monitor, the system displays the service operation data according to your most recent filtering selections.

   Note. In situations where multiple people are signing in with the same user ID, it is possible that their changes may collide with each other if more than one is refreshing the monitor pages at the same time. In such cases the system displays the message, 'Data updated by another user.'

Monitoring Asynchronous Service Operations

This section discusses how to:

- Filter asynchronous service operations data.
- Save asynchronous data filtering selections.
- View asynchronous filtering results.
- View monitor output for asynchronous service operations data.
- Monitor service operation transactions.
- Monitor asynchronous service operation instances.
- Monitor publication contracts.
- Monitor subscription contracts.

Understanding Monitoring Asynchronous Service Operations Data

This section provides an overview of asynchronous service operation statuses.

 Copyright © 1988-2007, Oracle. All rights reserved.

Asynchronous Service Operation Statuses

For asynchronous service operations, the Service Operations Monitor displays different statuses as service operations progress through the system.

The typical status progression for asynchronous service operations is:

1. *New*.
2. *Started*.
3. *Working*.
4. *Contracts created.*

However, the Service Operations Monitor can display any of the statuses listed in the following table.

Status	Description
New	Either the item has been written to the database but has not been dispatched yet, or the item has just been resubmitted.
Started	The dispatcher is in the process of passing the item to a handler, but the handler has not received it yet.
Working	The handler has accepted the item and is currently processing it.
Contracts Created	This status appears only for asynchronous service operations. It indicates that the service operation instance has completed processing.
Done	This status indicates different outcomes, depending on the type of process that you are monitoring. For publication contracts, the status indicates that publication has run successfully. For subscription contracts, the status indicates that the subscription has run successfully.
Error	An error occurred during processing. Manual intervention is required.
Retry	The system encountered an intermittent error during processing. The system retries service operations with this status automatically.
Timeout	The system has reached the maximum retry count to send a service operation.
Edited	The publication data for the item has been edited. Processing does not resume until you resubmit the item.

Copyright © 1988-2007, Oracle. All rights reserved.

Status	Description
Canceled	The item has been canceled. The system cannot process the item until you resubmit it.
Hold	This field is used in conjunction with message segmentation. The status of a segmented message is *Hold* while the system is processing the segments in the message.

Service Operation Status and Blocked and Stalled Queues

This section discusses:

- Blocked queues.
- Stalled queues.

Blocked Queues

Queues preserve the order in which service operations are processed.

The pub/sub system guarantees that items are processed in the order they are sent. If a service operation has a status of *Error*, *Timeout,* or *Edited,* the service operation queue becomes blocked and no processing occurs until you resolve the problem with the service operation.

For publications, queues are partitioned in queues by sub queues.

For publication contracts, the queues is further partitioned into queues by sub queue and target node. If a queue is ordered, items in that queue and in the same queue are processed in the order sent. The dispatcher does not begin processing an item until all items ahead of it in the queue have the status *Done* or *Cancelled*. An item with a status of *Error*, *Timeout,* or *Edited* blocks all items behind it in the same queue. If the remote node is unavailable, the dispatcher does not attempt to process the contract and the queue is blocked until the remote node becomes available. That is why publication contracts are partitioned by target node.

If a queue is unordered, an item (such as the publication, publication contract, or subscription contract) never blocks another item. All items are processed in parallel.

Stalled Queues

Stalls do not occur by design. They are caused by gaps in functionality, user errors, defects, and so forth.

For example, a queue can become stalled when:

- Multiple domains access the same database and one of the domains is shut down abnormally.

  Items may be stalled in the *Started* or *Working* status.

  Note. You can use the Domain Status page to correct the problem.

- A change occurs to the pub/sub runtime tables through direct SQL.

  The copies of the database tables that dispatchers have in memory are not updated. In this situation, you must reboot the dispatchers.

 Copyright © 1988-2007, Oracle. All rights reserved.

Pages Used to Monitor Asynchronous Service Operations

Page Name	Object Name	Navigation	Usage
Monitor Overview	IB_MONITOR_OVRVIEW	Select PeopleTools, Integration Broker, Service Operations Monitor, Monitoring, Asynchronous Services. Click the Monitor Overview tab.	Use this page to view items in the integration system at a high level so that you can isolate particular areas and then drill down for further information. You can select to group information by queue or service operation.
Operation Instances	IB_MONITOR_PUBHDR	Select PeopleTools, Integration Broker, Service Operations Monitor, Monitoring, Asynchronous Services. Click the Operation Instances tab.	View all asynchronous service operation instance from remote nodes or applications that publish information.
Publication Contracts	IB_MONITOR_PUBCON	Select PeopleTools, Integration Broker, Service Operations Monitor, Monitoring, Asynchronous Services. Click the Publication Contracts tab.	View outbound service operations sent to remote nodes with which the system is interacting.
Subscription Contracts	IB_MONITOR_SUBCON	Select PeopleTools, Integration Broker, Service Operations Monitor, Monitoring, Asynchronous Services. Click the Publication Contracts tab.	View inbound service operations data from remote nodes with which the system is interacting.
Sub Queue Operation Instances page	AMM_SUBCHNL_PUBHDR	• From the Asynchronous Services-Operation Instances page, click the link in the Sub Queue column in the Results grid. • From the Asynchronous Services-Subscription Contracts page, click the link in the Sub Queue column in the Results grid. • From the Asynchronous Services-Publication Contracts page, click the link in the Sub Queue column in the Results grid.	If sub queues exist for a queue, a Sub Queue Link appears in the Sub Queue column on the pages listed above. View all service operations in the sub queue in the order in which they will be processed. You can also resubmit service operations or cancel the submission of service operations on this page.

Filtering Asynchronous Service Operations Data

Use the following filter criteria to reduce your search results. The value you set on one page in the Asynchronous Services component is carried forward to all pages of this component. Unless stated otherwise, the fields display on all pages of the Asynchronous Services.

Copyright © 1988-2007, Oracle. All rights reserved.

Archived The Archived check box enables you to search for either archived or live service operation data. To search archived data, select the check box. To search live data, clear the check box.

External Service Name This field appears on the Operation Instances page in the Asynchronous Services component only.

Enter the name of the inbound service operation received from an integration partner. This name is equivalent to the routing alias.

Group By This field appears on the Monitor Overview page in the Asynchronous Services component only.

Use the dropdown list box to select how to group returned data. The valid values are:

- *Queue.* (Default.)

  Displays results by queue name.

- *Service Operation.*

  Displays results by service operation name.

Publish Node, Node Name Indicates the node that published the service operation.

Note. The Service Operations Monitor only allows you to view information for the local system (database). However, the queues for the local database can contain service operations published by remote nodes, as well the local node. There is only one local node for a database.

Queue Level This field appears on the Monitor Overview page in the Asynchronous Services component only.

The valid options are:

- *Oper Inst* (Operation Instance). (Default.)
- *Pub Con* (Publication Contract).
- *Sub Con* (Subscription Contract).

Queue Name To view service operation data within a specific queue, select the appropriate queue value in the Queue Name dropdown list box.

Refresh Click the button to apply the filtering criteria selected.

When you click the Refresh button the system saves your search criteria for subsequent searches.

See Chapter 21, "Using the Service Operations Monitor," Saving Filtering Selections, page 416.

Status To view service operation data by status, select the status criteria from the Status dropdown list box. The status options reflect the status columns that appear on the Monitor Overview page.

Descriptions of the possible service operation statuses are described elsewhere in this chapter.

See Chapter 21, "Using the Service Operations Monitor," Asynchronous Service Operation Statuses, page 417.

 Copyright © 1988-2007, Oracle. All rights reserved.

Time Period The Time Period group box features four fields for searching by date and time: From Date, To Date, From Time and To Time.

If you complete just the date fields, the time fields automatically populate from 12:01 a.m. to 11:59 p.m.

When left blank, no date or time is used as part of the search criteria.

Transaction ID To search for a specific transaction, enter the transaction ID.

On the pages where filtering applies, you enter your filtering criteria in the Message Criteria group box. The result set appears in the status grid directly below the filtering options.

Viewing Monitor Output for Asynchronous Service Operations Data

After you filter and search for asynchronous service operations data in the pages of the Asynchronous Services pages, the output displays in grid format at the bottom of the pages.

The following page elements display data related to integrations using asynchronous service operations in the grids:

Details Each row of filtering results on the Operation Instances page, Publication Contracts page and Subscription Contracts page displays a Details link. Click the link to view the data in the Asynchronous Details page, where you can view service operation properties, details about any service operation errors that have occurred, and view service operations in XML format.

Orig Trans ID The original transaction ID generated and used for the service operation instance.

As contracts are created another transaction ID is created for each publication or subscription contract. However, the original transaction ID is always available as a reference.

Publishing Node Name of the node sending node.

Queue Name The name of the queue used for the transaction.

Segment Number When implementing message segments, indicates the number of the segment message.

Send ID This field is used when ordered message segments are sent to a system that is not segment aware.

For example, say one message with three segments gets published and the queue sequence ID assigned by the system is 10.

If this message is sent to a target system that is not segment aware, the system breaks the message up into three separate messages and assigns each message a Send ID of *10*, *11* and *12* respectively before sending them to the target system.

Without the Send ID, the first message would be received, but the next two messages would be duplicates to the system because the IDs would be the same.

Service Operation Name of the service operation.

Service Operation Version Version of the service operation

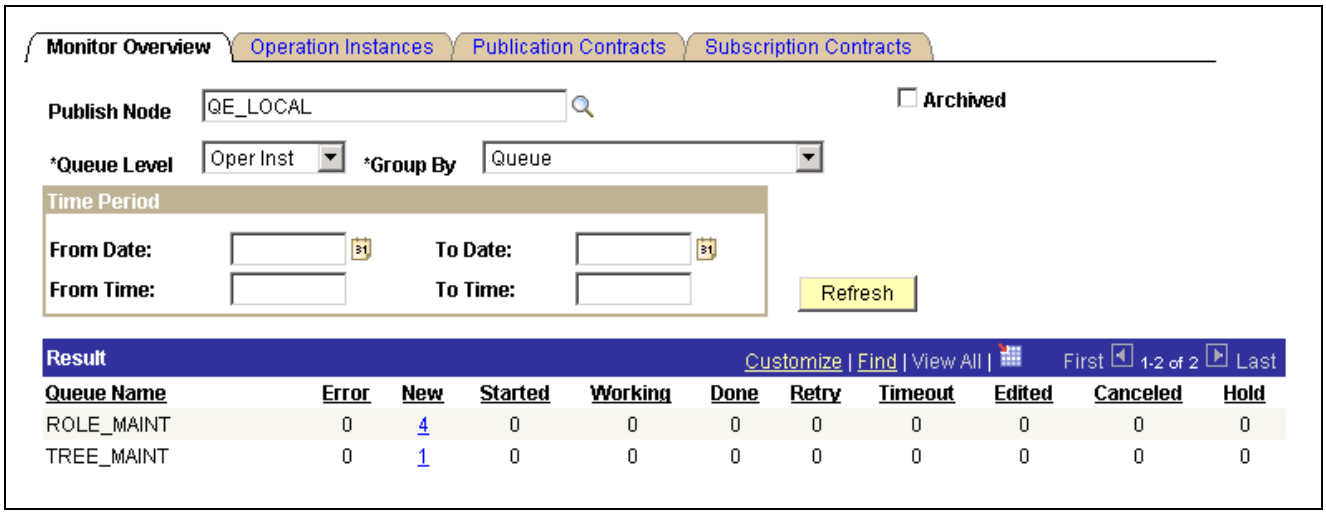
Copyright © 1988-2007, Oracle. All rights reserved.

Status	Status of the service operation in the system. Descriptions of the possible service operation statuses are described elsewhere in this chapter. See Chapter 21, "Using the Service Operations Monitor," Asynchronous Service Operation Statuses, page 417.
Subscriber Node	Name of the receiving node.
Sub Queue	If queue partitioning exists for a queue, a Sub Queue column appears in the Results grid on the Operation Instances page, Publication Contracts page and Subscription Contracts page . Click the link to open the Sub Queue Message Queue page to view all transactions in the sub queue.
Time Stamp	Date and time of the transaction.
Transaction ID	The unique identifier assigned to the transaction by the system.

Monitoring Service Operation Transactions

Use the Monitor Overview page for a high-level overview of the status of asynchronous service operation transactions. You can group transactions by queue or service operation for viewing.

To access this page, select PeopleTools, Integration Broker, Service Operations Monitor, Asynchronous Services. The Monitor Overview page shown in the following example appears:



Asynchronous Services-Monitor Overview page

After you search for queue information to view, the Results grid displays the results of your search.

This page displays search results by queue name or service operation name, depending on the selection you make in the Group By dropdown list box.

The processing status of service operations displays in the status columns (for example, *Error*, *New*, *Started*, and so on).

Copyright © 1988-2007, Oracle. All rights reserved.

Most of the time, the status for a service operation that appears in the Result grid is*Done*. This means that the service operation instance arrived in the publication queue (creating the service operation headers only). However, other statuses can appear. For instance, if the pub/sub system is down, the status is *New*. If there are transformation or PeopleCode errors, the service operation status is *Error*. In addition, if you access the Service Operations Monitor at certain times, you might see a status of *Started* or *Working*. Use the other pages in this component to view more comprehensive status information.

The preceding example shows that there are four service operation instances in *New* status in the *ROLE_MAIN* queue and one service operation instance in *New* status for the *TREE_MAINT* queue.

The number of operation instances in a particular status display as a linked value. Click the link to open the data in the Operation Instances page where you can view more detailed information.

See Also

Chapter 21, “Using the Service Operations Monitor,” Filtering Asynchronous Service Operations Data, page 419

Chapter 21, “Using the Service Operations Monitor,” Viewing Monitor Output for Asynchronous Service Operations Data, page 421

Chapter 21, “Using the Service Operations Monitor,” Asynchronous Service Operation Statuses, page 417

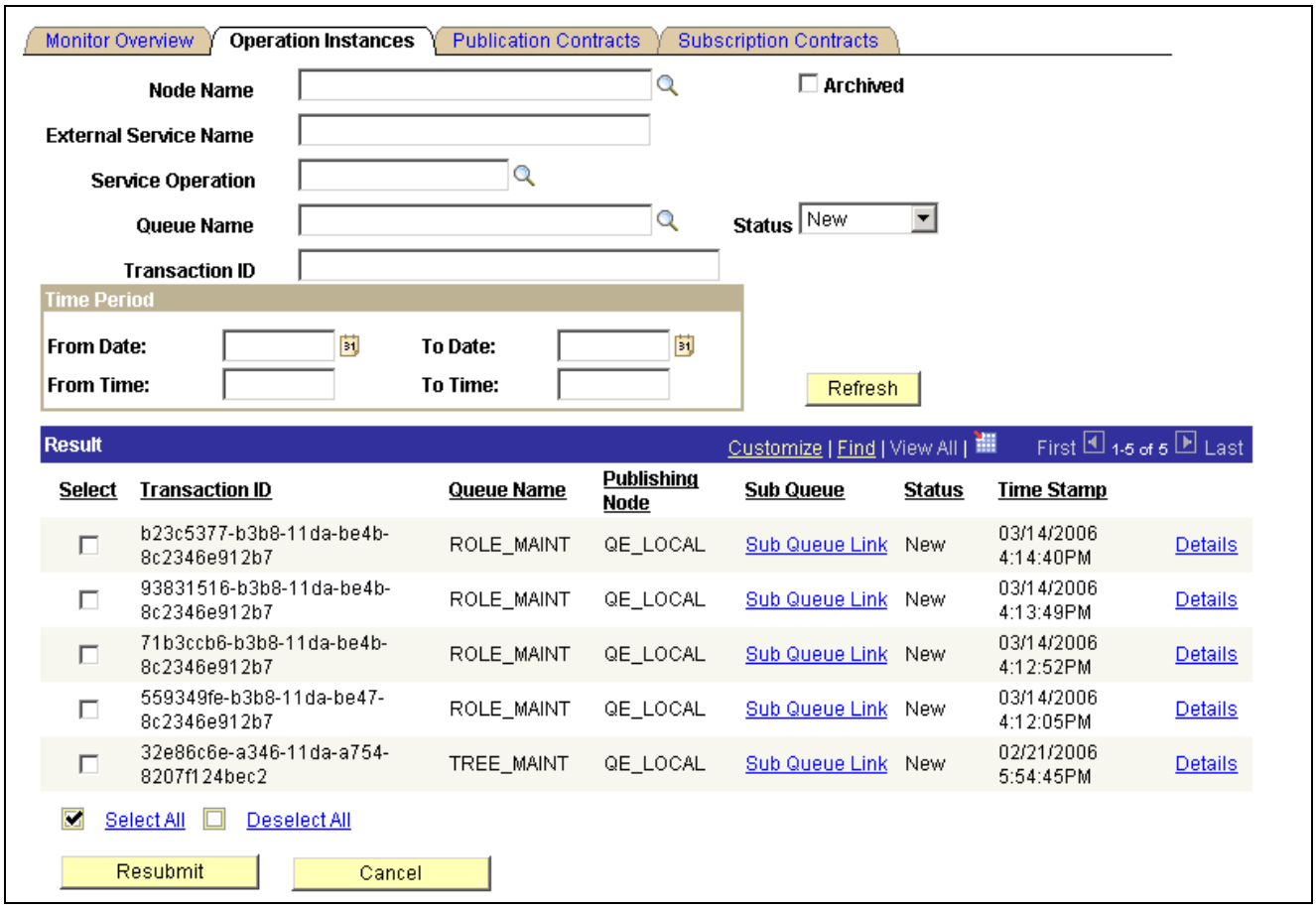
Chapter 21, “Using the Service Operations Monitor,” Monitoring Asynchronous Service Operation Instances, page 423

Monitoring Asynchronous Service Operation Instances

The Operation Instances page enables you to monitor the status and details related to individual asynchronous service operation instances.

To access the Operation Instances page, select PeopleTools, Integration Broker, Monitor Integrations, Monitor, Asynchronous Services and click the Operation Instances tab. The following page appears:

Copyright © 1988-2007, Oracle. All rights reserved.



Asynchronous Services-Operation Instances page

After you select your filtering options, click Refresh.

See Also

Chapter 21, “Using the Service Operations Monitor,” Filtering Asynchronous and Synchronous Service Operations Data, page 415

Chapter 21, “Using the Service Operations Monitor,” Viewing Monitor Output for Asynchronous Service Operations Data, page 421

Chapter 21, “Using the Service Operations Monitor,” Asynchronous Service Operation Statuses, page 417

Chapter 21, “Using the Service Operations Monitor,” Viewing Asynchronous Service Operation Details, page 427

Chapter 21, “Using the Service Operations Monitor,” Resubmitting and Canceling Service Operations for Processing, page 440

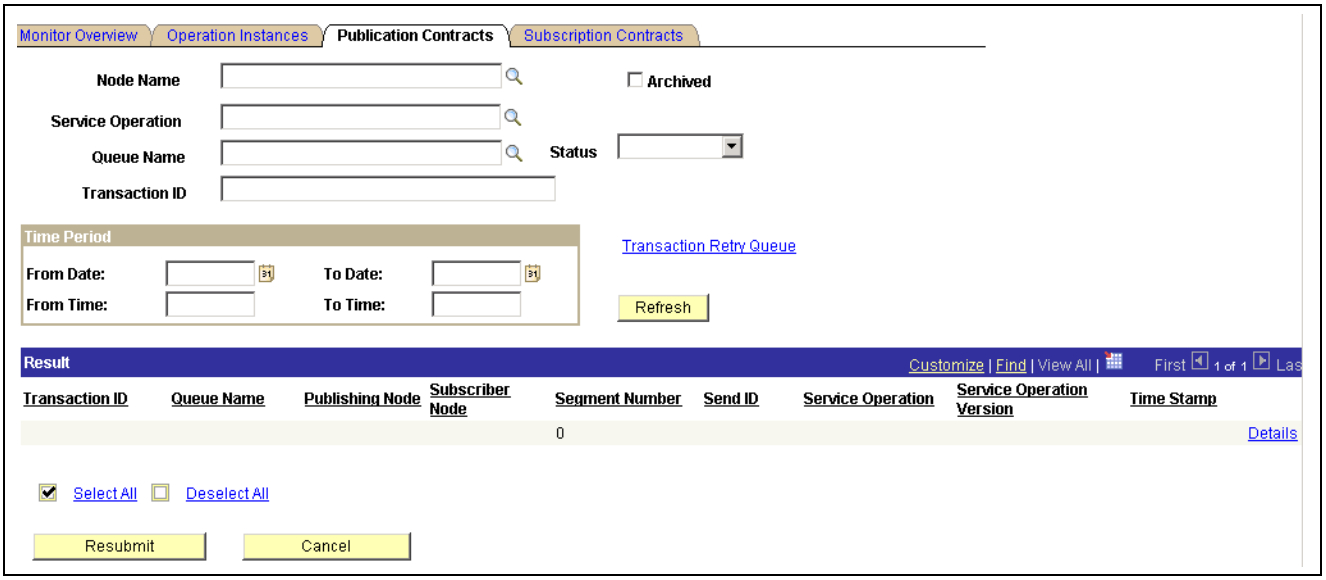
Chapter 21, “Using the Service Operations Monitor,” Viewing Queue Partitioning Information, page 426

Monitoring Publication Contracts

The Publication Contracts page shows outbound publication transactions to send to remote nodes.

 Copyright © 1988-2007, Oracle. All rights reserved.

To access the page, select PeopleTools, Integration Broker, Service Operations Monitor, Monitoring, Asynchronous Services. Then click the Publication Contracts tab. The following example shows the Publication Contracts page:



Asynchronous Service-Publication Contracts page

The system does not create publication contracts for routing to the local node.

See Also

Chapter 21, “Using the Service Operations Monitor,” Filtering Asynchronous and Synchronous Service Operations Data, page 415

Chapter 21, “Using the Service Operations Monitor,” Viewing Monitor Output for Asynchronous Service Operations Data, page 421

Chapter 21, “Using the Service Operations Monitor,” Asynchronous Service Operation Statuses, page 417

Chapter 21, “Using the Service Operations Monitor,” Viewing Asynchronous Service Operation Details, page 427

Chapter 21, “Using the Service Operations Monitor,” Resubmitting and Canceling Service Operations for Processing, page 440

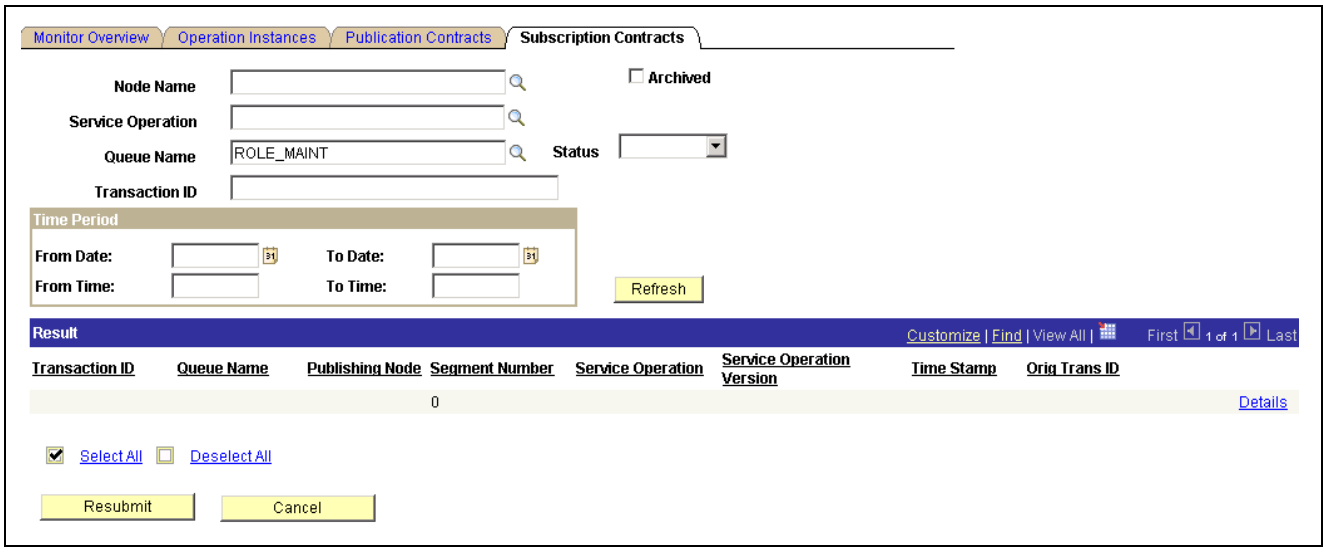
Chapter 21, “Using the Service Operations Monitor,” Viewing Queue Partitioning Information, page 426

Monitoring Subscription Contracts

The Subscription Contracts page enables you to view transactions to which the local node subscribes. Subscription contracts for remote nodes do not appear.

To access this page, select PeopleTools, Integration Broker, Service Operations Monitor, Monitoring, Asynchronous Services. Then click the Subscription Contracts tab. The following example shows the Subscription Contracts page:

Copyright © 1988-2007, Oracle. All rights reserved.



Asynchronous Services-Subscription Contracts page

Note. When viewing the status of bulk subscription contracts (such as 100,000 or more) using a Solaris operating system and an Oracle database, your browser session may close unexpectedly. As a result, you should filter the number of subscription contracts for which to view status information. To do so, use the settings in the Time Period box to filter information by date and time. The volume of service operations in the system determines the best values to enter.

See Also

Chapter 21, "Using the Service Operations Monitor," Filtering Asynchronous and Synchronous Service Operations Data, page 415

Chapter 21, "Using the Service Operations Monitor," Viewing Monitor Output for Asynchronous Service Operations Data, page 421

Chapter 21, "Using the Service Operations Monitor," Asynchronous Service Operation Statuses, page 417

Chapter 21, "Using the Service Operations Monitor," Viewing Asynchronous Service Operation Details, page 427

Chapter 21, "Using the Service Operations Monitor," Resubmitting and Canceling Service Operations for Processing, page 440

Chapter 21, "Using the Service Operations Monitor," Viewing Queue Partitioning Information, page 426

Viewing Queue Partitioning Information

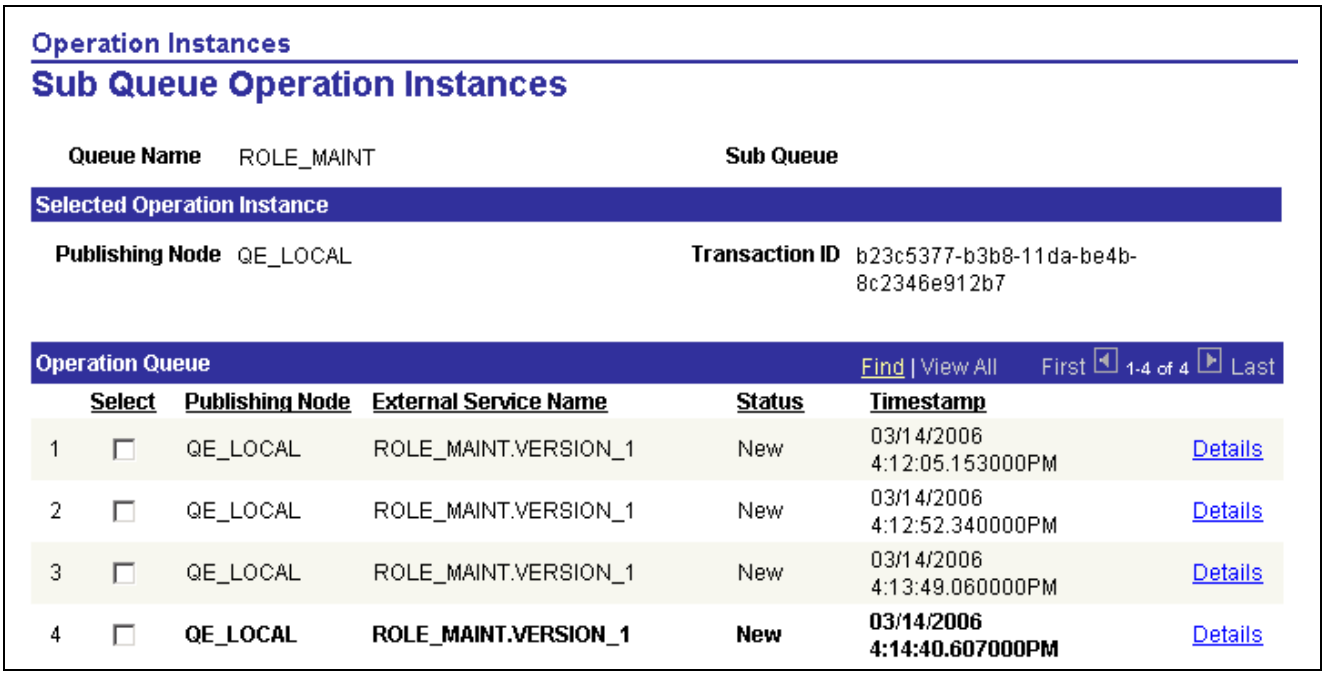
If queue partitioning exists for a queue, a Sub Queue column appears in the Results grid on the Monitor Overview page, the Publication Contracts page, and the Subscription Contracts page of the Asynchronous Services component.

If you click the sub queue name link, the Sub Queue Operation Instances page displays and you can view all service operations in the sub queue in the order in which they will be processed. You can also resubmit service operations or cancel the submission of service operations on this page.

Note. When viewing sub queue info, even if the primary page was displaying archived data, this page always shows current data.

Copyright © 1988-2007, Oracle. All rights reserved.

The following example shows the Sub Queue Operation Instances page:



Sub Queue Operation Instances page

The row in bold is the row you were viewing on the previous page.

Descriptions of the page elements that appear on the page are described elsewhere in this section.

See Chapter 21, "Using the Service Operations Monitor," Viewing Monitor Output for Asynchronous Service Operations Data, page 421.

Viewing Asynchronous Service Operation Details

The Asynchronous Details component enables you to gather in-depth information about a specific asynchronous service operation. It also enables you to perform tasks such as correct errors and resubmit service operations.

This section discusses how to:

- View asynchronous service operation instance details.
- View asynchronous publication contract details.
- View asynchronous subscription contract details

Common Elements Used to View Asynchronous Service Operation Details

Cancel Click the Cancel button to cancel processing attempts for a service operation. This button is enabled when a service operation has a status of *New*, *Retry*, *Time Out*, *Error*, or *Edited*.

Copyright © 1988-2007, Oracle. All rights reserved.

Edit XML
An Edit XML link appears when there are errors with the transaction. Click the link to edit the service operation instance, publication contract or subscription contract XML to correct errors.

If you do not have appropriate permission for the particular service operation being viewed, this link is disabled.

Error Messages
This link can appear in the service operation instance details section, the publication contracts section, or the subscription contracts section.

Click the link to view error messages for these items.

If the link is disabled, there are no errors to view or you do not have the appropriate permissions to view the information.

Last Update Date/Time
Displays the date and time the transaction was last updated.

Process Identifier
Identifies the process ID on the local application server.

Resubmit
Click the Resubmit button to resubmit a service operation for processing. This button is enabled when a service operation has a status of *Time Out*, *Error*, *Edited*, or *Cancelled*. If a service operation contains an error or has timed out, typically you can just correct the problem and resubmit the service operation. After you edit a service operation, the status becomes *Edited.* When you resubmit the service operation, the status changes, yet again, to *New*.

If you do not have appropriate permission for the particular service operation being viewed, this button is disabled.

Retry Count
If the first attempt to deliver the service operation failed, this value reflects the number of times the system has attempted to resend the service operation.

Segment
If using message segments, indicates the segment number for which the page or section is displaying information.

When working with asynchronous operation instance details, use the Segment dropdown list box to select a different segment for which to view information. Click the Refresh button to refresh the page.

Service Operation
Indicates the name of the service operation.

Status
Status of the service operation in the system.

Descriptions of the possible service operation statuses are described elsewhere in this chapter.

Transaction ID
Displays the unique identifier that the system assigns to each transaction.

Transaction Type
Indicates the transaction type. The valid values are:

- *Inbound synchronous.*
- *Outbound synchronous.*

Version(Service operation)
Indicates the service operation version.

View IB Info
Click the link to view IB info.

If you do not have appropriate permission for the particular service operation being viewed, this link is disabled.

Copyright © 1988-2007, Oracle. All rights reserved.

View XML Click to view XML for the service operation instance, publication contract or subscription contract.

If you do not have appropriate permission for the particular service operation being viewed, this link is disabled.

See Also

Chapter 21, “Using the Service Operations Monitor,” Resubmitting and Canceling Service Operations for Processing, page 440

Chapter 21, “Using the Service Operations Monitor,” Viewing Service Operation IB Info Data, page 442

Chapter 21, “Using the Service Operations Monitor,” Viewing Service Operation Errors, page 443

Chapter 21, “Using the Service Operations Monitor,” Viewing and Editing Service Operation XML, page 446

Chapter 21, “Using the Service Operations Monitor,” Viewing Service Operation Nonrepudiation Signature Information, page 449

Copyright © 1988-2007, Oracle. All rights reserved.

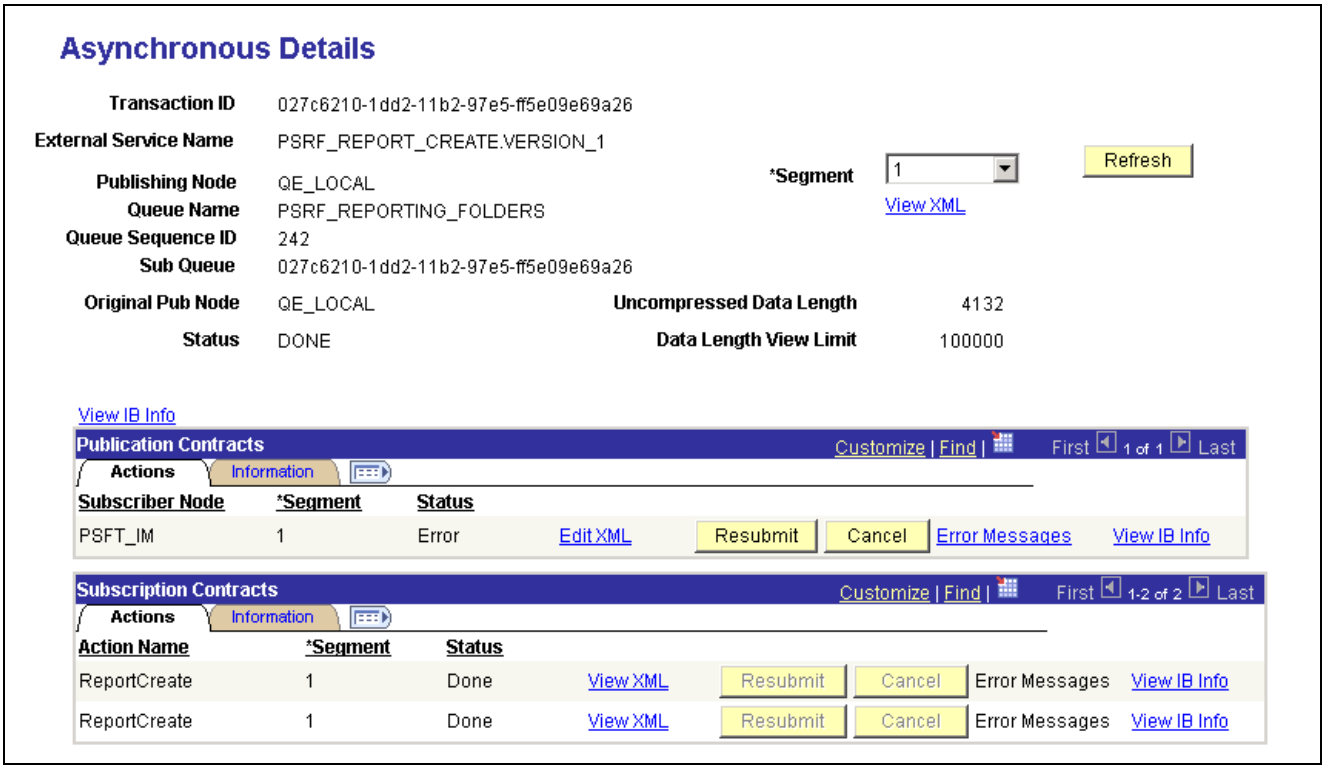
Pages Used to View Asynchronous Service Operation Details

Page Name	Object Name	Navigation	Usage
Asynchronous Details	IB_MONITOR_DET	You can use the following methods to access the Asynchronous Details page: • Select PeopleTools, Integration Broker, Service Operations Monitor, Monitoring, Asynchronous Details. • Select PeopleTools, Integration Broker, Service Operations Monitor, Monitoring, Asynchronous Services and click the Operation Instances tab. From any result row in the Results grid, click the Details link. • Select PeopleTools, Integration Broker, Service Operations Monitor, Monitoring, Asynchronous Services and click the Publication Contracts tab. From any result row in the Results grid, click the Details link. • Select PeopleTools, Integration Broker, Service Operations Monitor, Monitoring, Asynchronous Services and click the Subscription Contracts tab. From any result row in the Results grid, click the Details link.	View asynchronous service operation properties. View and correct service operation errors, and resubmit, cancel, archive and delete service operations. View asynchronous service operations in XML format. View nonrepudiation and IB info information.
Monitor Options Setup page	IB_MONITORADMIN	Select PeopleTools, Integration Broker, Service Operations Monitor, Administration, Monitor Options Setup.	Set the data view length limit for service operation XML. The data view length limit determines the size of service operation XML (in bytes) that is automatically loaded into XML Viewer for viewing in the Asynchronous Details component.

Copyright © 1988-2007, Oracle. All rights reserved.

Viewing Asynchronous Service Operation Instance Details

Use the Asynchronous Details page to view asynchronous service operation instance details. To access the page, select PeopleTools, Integration Broker, Service Operations Monitor, Monitoring, Asynchronous Details. The following example shows the Asynchronous Details page:



Asynchronous Details page

The section at the top of the Asynchronous Details page provides general information pertaining to a particular service operation instance to assist in troubleshooting.

Note. The data and fields that display in the Publication Contracts and Subscription Contracts section of the page are discussed elsewhere in this section.

External Service Name Indicates the name of the service operation sent by the sending node.

Publishing Node Identifies the name of the sending node.

Queue Name Identifies the queue to which the service operation is associated.

Queue Sequence ID Identifies the sequence of a particular service operation in the a queue.

Sub Queue If queue partitioning exists for a queue, indicates the name of the sub queue to which the service operation is associated.

Original Pub Node Indicates the name of the original sending node.

In most cases the original publishing node and the publishing node are the same. However, if the service operation goes through a hub, the original publishing node and publishing node differ.

Refresh Click the button to refresh page data.

Copyright © 1988-2007, Oracle. All rights reserved.

Archive	Click the Archive button to archive a service operation. This button is enabled when a service operation has a status of *Done* or *Cancelled* and no associated contract has pending work. If the queue is not set up for archiving, the Archive button is replaced with a Delete button. .
Uncompressed Data Length	Indicates the size of the XML service operation in bytes.
Data Length View Limit	Indicates the maximum size of an XML document in bytes that is automatically loaded in the XML Viewer page. The default is *100000* bytes. Set this property in the Service Operations Monitor using the Monitor Setup Options page. See Chapter 21, "Using the Service Operations Monitor," Setting the Data Length View Limit for Displaying XML, page 434.

Other page elements that appear on the page are discussed elsewhere in this section.

See Also

Chapter 21, "Using the Service Operations Monitor," Common Elements Used to View Asynchronous Service Operation Details, page 427

Chapter 21, "Using the Service Operations Monitor," Resubmitting and Canceling Service Operations for Processing, page 440

Chapter 21, "Using the Service Operations Monitor," Viewing Service Operation IB Info Data, page 442

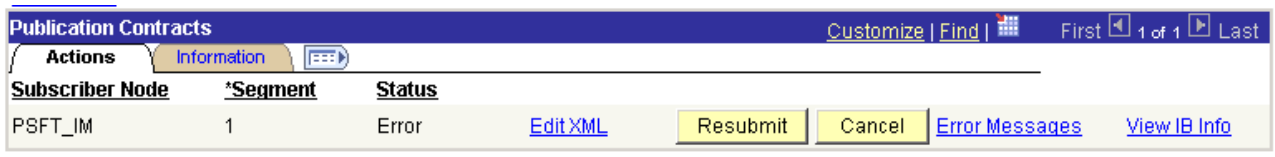
Chapter 21, "Using the Service Operations Monitor," Viewing Service Operation Errors, page 443

Chapter 21, "Using the Service Operations Monitor," Viewing and Editing Service Operation XML, page 446

Chapter 21, "Using the Service Operations Monitor," Viewing Service Operation Nonrepudiation Signature Information, page 449

Viewing Asynchronous Publication Contracts Details

Use the Publication Contracts section of the Asynchronous Details page to view asynchronous publication contract details. The following example shows this section:



Asynchronous Details-Publication Contracts section.

Note. The section displays only when there are publication contracts associated with the service operation.

Viewing and Working with Publication Actions

The Actions tab reveals all the nodes subscribing to a particular service operation and the current status of the publication contract, as in whether the publication has been successfully posted to the subscribing node.

The Actions tab in the Publication Contracts section provides the following information.

Copyright © 1988-2007, Oracle. All rights reserved.

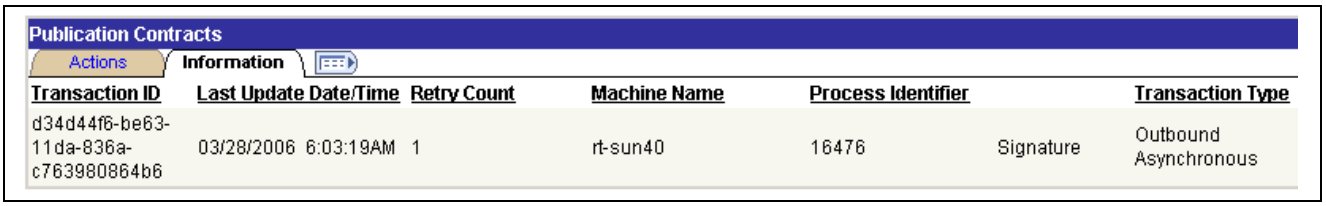
Subscriber Node Identifies the name of the subscribing or receiving node.

Other page elements that appear on the page are discussed elsewhere in this section.

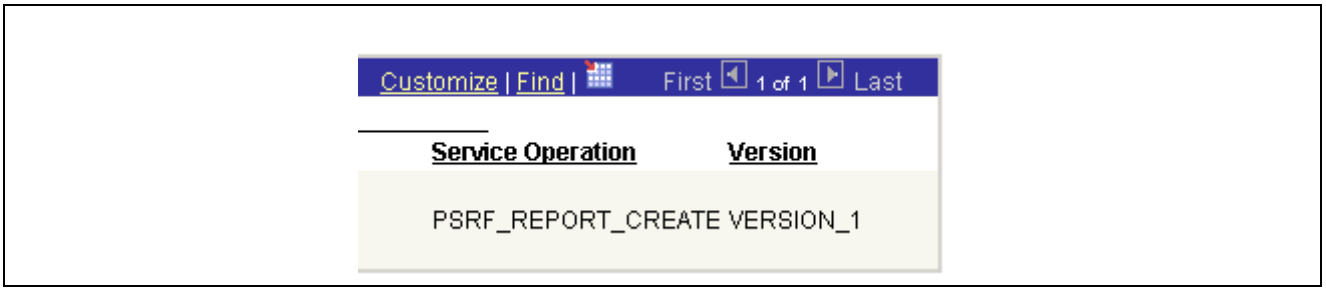
See Chapter 21, "Using the Service Operations Monitor," Common Elements Used to View Asynchronous Service Operation Details, page 427; Chapter 21, "Using the Service Operations Monitor," Resubmitting and Canceling Service Operations for Processing, page 440; Chapter 21, "Using the Service Operations Monitor," Viewing Service Operation IB Info Data, page 442; Chapter 21, "Using the Service Operations Monitor," Viewing Service Operation Errors, page 443 and Chapter 21, "Using the Service Operations Monitor," Viewing and Editing Service Operation XML, page 446.

Viewing Publication Information Details

The Information tab reveals details about the publication transaction, including the transaction ID, the transaction time stamp, and so on. The following examples show the Information tab:



Publication Contracts section-Information tab (continued.)



(Continued.) Publication Contracts-Information tab

The Information tab contains the following information about the publication contract:

Machine Name Identifies the name of the local application server machine that processed the publication contract.

Signature When nonrepudiation is implemented, this page element displays as a hyperlink.

Click the link to view nonrepudiation information associated with the publication contract.

Other page elements that appear on the page are discussed elsewhere in this section.

See Chapter 21, "Using the Service Operations Monitor," Common Elements Used to View Asynchronous Service Operation Details, page 427 and Chapter 21, "Using the Service Operations Monitor," Viewing Service Operation Nonrepudiation Signature Information, page 449.

Viewing Asynchronous Subscription Contracts Details

Use the Subscription Contracts section of the Asynchronous Details page to view asynchronous subscription contract details. The following example shows this section:

Copyright © 1988-2007, Oracle. All rights reserved.

Note. The section displays only when there are subscription contracts associated with the service operation.

Viewing and Working with Subscription Actions

The Actions tab of the Subscription Contracts section of the Asynchronous Services page reveals the status of a particular subscription contract.

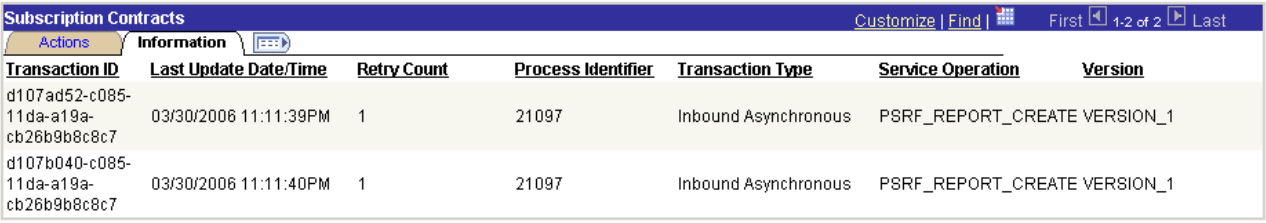
Publication Contracts section of the Asynchronous Details page.

Note. The page elements that appear on the page are discussed elsewhere in this section.

See Chapter 21, "Using the Service Operations Monitor," Common Elements Used to View Asynchronous Service Operation Details, page 427; Chapter 21, "Using the Service Operations Monitor," Resubmitting and Canceling Service Operations for Processing, page 440; Chapter 21, "Using the Service Operations Monitor," Viewing Service Operation IB Info Data, page 442; Chapter 21, "Using the Service Operations Monitor," Viewing Service Operation Errors, page 443 and Chapter 21, "Using the Service Operations Monitor," Viewing and Editing Service Operation XML, page 446.

Viewing and Working with Subscription Information

The Information tab reveals details about the subscription transaction, including the transaction ID, the transaction time stamp, and so on. The following examples show the Information tab:



Subscription Contracts section-Information tab

Note. The page elements that appear on the page are discussed elsewhere in this section.

See Chapter 21, "Using the Service Operations Monitor," Common Elements Used to View Asynchronous Service Operation Details, page 427.

Setting the Data Length View Limit for Displaying XML

The data view length limit determines the size of service operation XML (in bytes) that is automatically loaded into the XML Viewer in the Asynchronous Details component.

The default is *100000* bytes.

If the limit is exceeded, you are given the option of downloading and uploading the XML to view it or make changes.

Copyright © 1988-2007, Oracle. All rights reserved.

You can change the default value using the Monitor Setup Options page in the Services Operations Monitor.

To set the data length view limit:

1. Select PeopleTools, Integration Broker, Service Operations Monitor, Administration, Monitor Setup Options.

   The Monitor Setup Options page appears.

2. In the Data Length View Limit box, enter a value in bytes.

   Note. Do not enter a negative value.

   Click the Save button.

Monitoring Synchronous Service Operations

This section discusses how to:

- Filter synchronous service operations data.
- View monitor output for synchronous service operations data.

Understanding Synchronous Service Operation Statuses

For synchronous service operations, the Service Operations Monitor displays the following statuses as synchronous service operations progress through the integration system:

Status	Description
Done.	Indicates the synchronous request was successful.
Error.	Indicates that an error occurred during processing. Manual intervention is required.

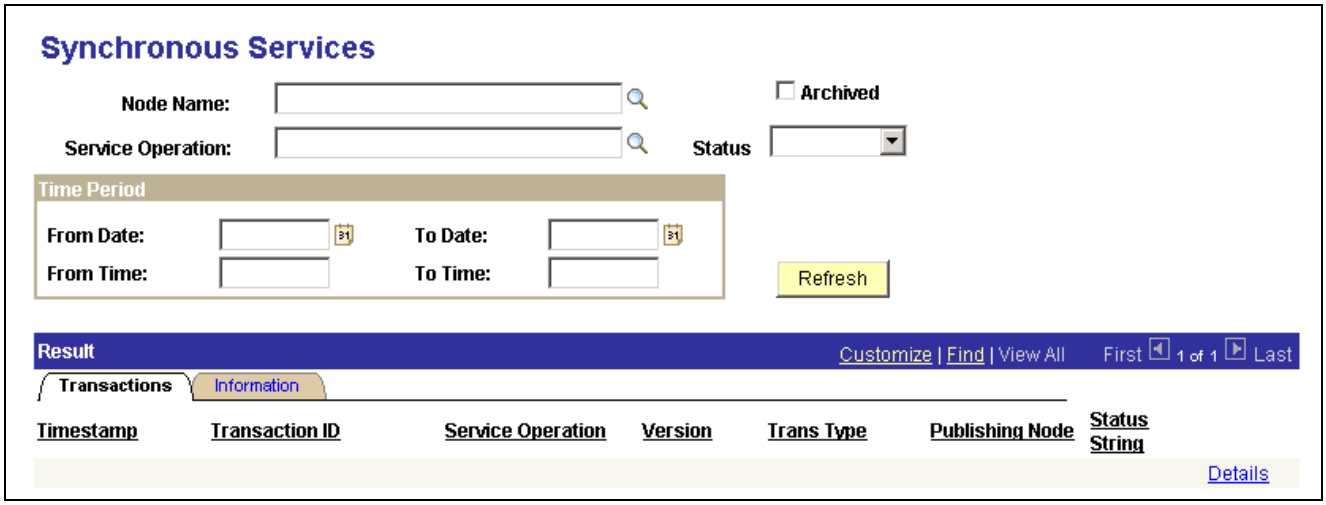
Page Used to View Synchronous Service Operations

Page Name	Object Name	Navigation	Usage
Synchronous Services	AMM_SYNCMSGLIST	Select PeopleTools, Integration Broker, Service Operations Monitor, Monitoring, Synchronous Services	Use the page to view data about inbound synchronous service operations from remote nodes or application that publish information, such as transaction ID, service operation and version, publishing node, and so on.

Copyright © 1988-2007, Oracle. All rights reserved.

Filtering Synchronous Service Operations Data

Use the Synchronous Services page to filter and view inbound synchronous service operations data in the integration system. To access the page, select PeopleTools, Integration Broker, Service Operations Monitor, Monitoring, Synchronous Services. The following example shows the Synchronous Services component:



Synchronous Services page

Use the following filter criteria when working with the Synchronous Services page to reduce your search results.

Node Name Identifies the name of the sending node.

Service Operation Identifies the name of the service operation for which to view data.

Archived The Archived check box enables you to search for either archived or live service operation data. To search archived data, select the check box. To search live data, clear the check box.

Status To view service operation data by status, select the status criteria from the Status dropdown list box. The status options reflect the status columns that appear on the Monitor Overview page.

Descriptions of the possible service operation statuses are described elsewhere in this chapter.

Refresh Click the button to apply the filtering criteria selected.

When you click the Refresh button the system saves your search criteria for subsequent searches.

Time Period The Time Period group box features four fields for searching by date and time: From Date, To Date, From Time and To Time.

When left blank, no date or time is used as part of the search criteria. If only the date fields are populated, the system automatically fills in the time fields.

See Also

Chapter 21, “Using the Service Operations Monitor,” Archiving Service Operation Instances, page 453

Chapter 21, “Using the Service Operations Monitor,” Understanding Synchronous Service Operation Statuses, page 435

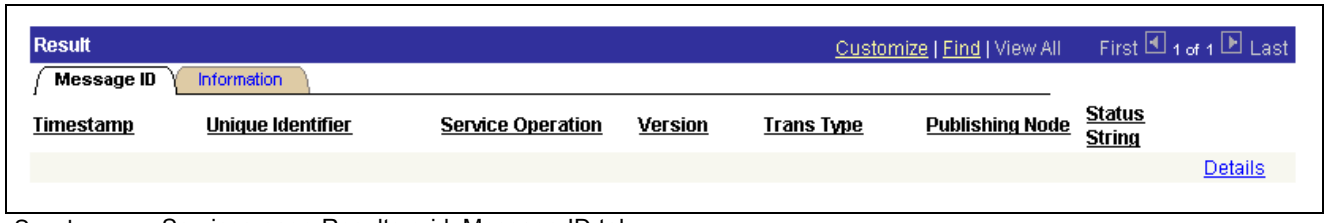
 Copyright © 1988-2007, Oracle. All rights reserved.

Viewing Monitor Output for Synchronous Service Operations Data

After you filter and search for synchronous service operations data on the Synchronous Services page, the output displays in a Results grid at the bottom of the page.

Viewing Synchronous Service Operation Message ID Information

The following example shows the Message ID tab of the Results grid on the Synchronous Services page.



Synchronous Services page Results grid–Message ID tab

You can view the following data in the section:

Timestamp Identifies the date and time that the service operation instance was last processed.

Unique Identifier Displays the transaction ID, the unique identifier that the system assigns to each transaction.

Service Operation Indicates the name of the service operation.

Version Indicates the version of the service operation.

Trans Type Identifies the transaction type. Values are:

- *OutSync*: Outbound Synchronous
- *InSync*: Inbound Synchronous

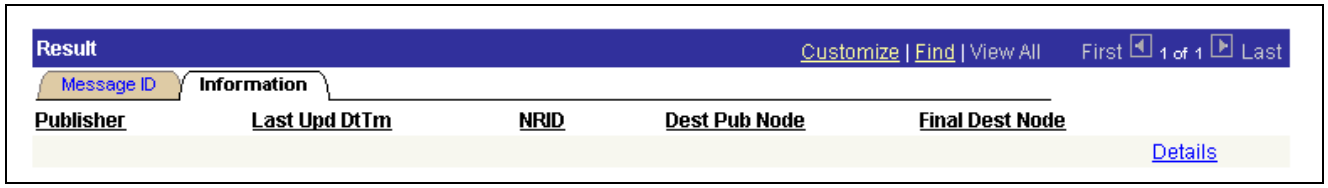
Publishing Node Indicates the sending node.

Status String Indicates the status of the service operation.

Details Click the link to open the Synchronous Details page for the service operation to view more in-depth data about the transaction.

Viewing Synchronous Service Operation General Information

The following example shows the Information tab of the Results grid on the Synchronous Services page.



Synchronous Services page Results grid–Information tab

You can view the following data in the section:

Publisher Indicates the name of the sending node

Copyright © 1988-2007, Oracle. All rights reserved.

Last Upd Dt Tm	Indicates the date and time the transaction was last updated.
NRID(Nonrepudiation ID)	Displays when nonrepudiation is implemented. Identifies a unique number used to associate a service operation instance with the nonrepudiation log.
Dest Pub Node	Identifies the name of the node where the service operation will be sent.
Final Dest Node	Identifies the name of the node of the final destination for the service operation.
Details	Click the link to open the Synchronous Details page for the service operation to view more in-depth data about the transaction.

Viewing Synchronous Service Operation Instance Details

This section discusses how to synchronous service operation details.

Pages Used to View Synchronous Service Operations Instance Details

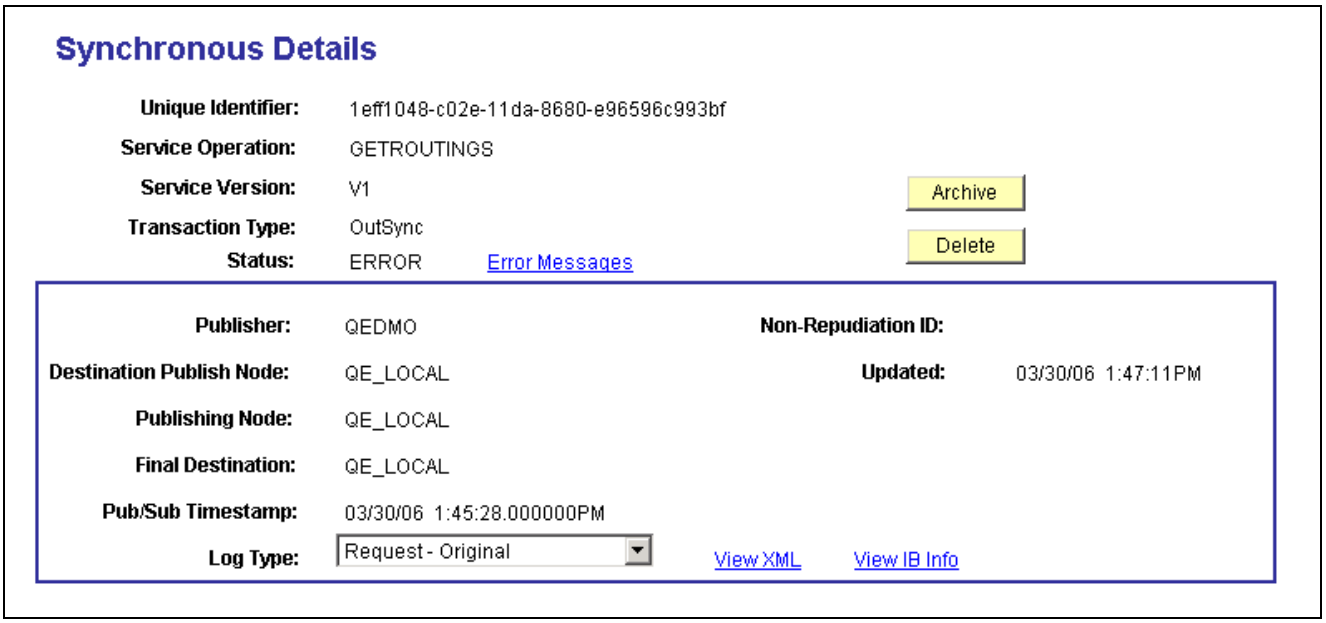
Page Name	Object Name	Navigation	Usage
Synchronous Details page	AMM_SYNCDETAIL	Select PeopleTools, Integration Broker, Service Operations Monitor, Monitoring, Synchronous Details. • Select PeopleTools, Integration Broker, Service Operations Monitor, Monitoring, Synchronous Details. • Select PeopleTools, Integration Broker, Service Operations Monitor, Monitoring, Synchronous Services. Click the Details link in Message ID tab of the Results grid. • Select PeopleTools, Integration Broker, Service Operations Monitor, Monitoring, Synchronous Services. Click the Details link in Information tab of the Results grid.	Use the page to gather in-depth information about a specific synchronous service operation. It also enables you to perform tasks such as view logs, and archive and delete transactions.

Viewing Synchronous Service Operation Details

The Synchronous Detail page provides read-only information about synchronous service operations in the system. It also enables you to view signature information for a service operation if it was processed with nonrepudiation logic.

Copyright © 1988-2007, Oracle. All rights reserved.

To access the page, select PeopleTools, Integration Broker, Service Operations Monitor, Monitoring, Synchronous Services. The following example shows the Synchronous Details page.



Synchronous Details page

The page displays data in the following page elements:

Archive Click the button to archive the synchronous service operation.

Delete Click the delete button to delete the transaction from the database.

Destination Publish Node Identifies the name of the node where the service operation was sent.

Error Messages Click the link to view error messages associated with the processing of the service operation.

Final Destination Identifies the name of the node of the final destination for the service operation.

Log Type Select a value from the dropdown list box and click the View XML link to view the corresponding information.

Note. For synchronous service operations, to view full service operation details in XML you must set a parameter in the routing definition for the service operation. On the Routing-Routing Definitions page, from the Log Detail dropdown list box select *Header and Detail*.

Values are:

- *Request Original*: Displays the original request data in XML format.
- *Request Transformed*: Displays transformed request data, if applicable, in XML format.
- *Response Original*: Displays the original response data in XML format.
- *Response Transformed*: Displays the transformed response data, if applicable, in XML format.

Copyright © 1988-2007, Oracle. All rights reserved.

Non-Repudiation ID	Identifies a unique number used to associate a service operation instance with the nonrepudiation log.
Publisher	Publisher of the service operation. This is usually the user ID of the person in the publishing system who triggered the publication.
Pub/Sub Timestamp	Identifies the date and time that the service operation instance was last processed.
Service Operation	Identifies the name of the service operation published.
Service Version	Identifies the version of the service operation published.
Signature	If a service operation is sent with a signature, a Signature link appears next to the Non-Repudiation ID field. When you click the Signature link, the service operation signature appears in XML format.
Status	Identifies the status of the service operation. Descriptions of the possible service operation statuses are described elsewhere in this chapter.
Transaction Type	Identifies the transaction type. Values are: • *OutSync*: Outbound Synchronous • *InSync*: Inbound Synchronous
Unique Identifier	Displays the transaction ID, the unique identifier that the system assigns to each transaction.
Updated	Identifies the date and time the service operation was last updated.
View IB Info	Click the link to view IB info in XML format for the service operation, such as transaction ID,
View XML	Click to view the service operation content in XML format.

See Also

Chapter 21, “Using the Service Operations Monitor,” Viewing and Editing Service Operation XML, page 446

Chapter 21, “Using the Service Operations Monitor,” Viewing Service Operation IB Info Data, page 442

Chapter 21, “Using the Service Operations Monitor,” Viewing Service Operation Nonrepudiation Signature Information, page 449

Chapter 21, “Using the Service Operations Monitor,” Resubmitting and Canceling Service Operations for Processing, page 440

Chapter 21, “Using the Service Operations Monitor,” Understanding Synchronous Service Operation Statuses, page 435

Resubmitting and Canceling Service Operations for Processing

This section discusses how to:

 Copyright © 1988-2007, Oracle. All rights reserved.

- Resubmit or cancel individual service operations for processing.
- Resubmit or cancel service operations for processing in bulk.

Understanding Resubmitting and Canceling Service Operations for Processing

You can resubmit and cancel service operations only for those to which you have permissions. If you attempt resubmit or cancel a service operation for which you do not have permission, the system ignores the action.

Pages Used to Resubmit and Cancel Service Operations for Processing

Page Name	Object Name	Navigation	Usage
Operation Instances	IB_MONITOR_PUBHDR	Select PeopleTools, Integration Broker, Service Operations Monitor, Asynchronous Services. Click the Operation Instances tab.	Select one or more service operations to resubmit or cancel in the results grid and click the appropriate Resubmit or Cancel button to perform the action.
Publication Contracts	IB_MONITOR_PUBCON	Select PeopleTools, Integration Broker, Service Operations Monitor, Asynchronous Services. Click the Publication Contracts tab.	Select one or more service operations to resubmit or cancel in the results grid and click the appropriate Resubmit or Cancel button to perform the action.
Subscription Contracts	IB_MONITOR_SUBCON	Select PeopleTools, Integration Broker, Service Operations Monitor, Monitoring, Asynchronous Services. Click the Subscription Contracts tab.	Select one or more service operations to resubmit or cancel in the results grid and click the appropriate Resubmit or Cancel button to perform the action.
Asynchronous Details page	IB_MONITOR_DET	Select PeopleTools, Integration Broker, Service Operations Monitor, Monitoring, Asynchronous Details	On the Publication Contracts or Subscription Contracts section Actions tab, click the Cancel or Resubmit button to cancel or resubmit individual publication contracts or subscription contracts.

Resubmitting and Canceling Individual Service Operations

To resubmit or cancel individual service operations, select the check box next to the appropriate service operation and click the Resubmit or Cancel button.

To deselect a service operation, clear the check box next to the service operation.

Copyright © 1988-2007, Oracle. All rights reserved.

Resubmitting and Canceling Service Operations in Bulk

In addition to the Clear All, Resubmit and Cancel buttons, you can also use the following links when resubmitting and canceling service operations in bulk.

Select All Click to select all service operations in the results grid for resubmittal or cancellation. After you click this link, click the Resubmit or Cancel button as appropriate.

Deselect All Click the link to deselect all service operations in the results grid.

Viewing Service Operation IB Info Data

This section discusses how to view IB info data.

Pages Used to View IB Info Data

Page Name	Object Name	Navigation	Usage
View IB Info page	AMM_IBINFO	Select PeopleTools, Integration Broker, Service Operations Monitor, Monitoring, Asynchronous Details.Click the View IB Info link in any of the following locations: • Under the service operation details to view IB info for the service operation instance. • In the Publication Contracts section to view IB info data for a publication contract. • In the Subscription Contracts section to view IB info data for a subscription contract.	View IB info for an asynchronous service operation, publication contract or subscription contract. View IB Info data for a synchronous service operation.
View IB Info page	AMM_IBINFO	Select PeopleTools, Integration Broker, Service Operations Monitor, Monitoring, Synchronous Details.	View IB info for a synchronous service operation instance.

Viewing IB Info Data

A View IB Info link appears in several locations on the Asynchronous Details page and enables you to view IB info data for asynchronous service operations instances, publication contracts and subscription contracts. In addition, a View IB Info link displays on the Synchronous Details page and enables you to view IB info data for synchronous service operation as well.

Copyright © 1988-2007, Oracle. All rights reserved.

When you click the View IB Info link, the View IB Info page appears and displays information such as requesting node, transaction ID, content type, and so on.

The following example shows the View IB Info page:

View IB Info page

When you are done reviewing the data, click the Return button to return to the previous page.

Viewing Service Operation Errors

This section discusses how to:

- View asynchronous service operation instance errors.
- View publication contract errors.
- View subscription contract errors.
- View subscription service operation instance errors.

Common Elements Used in This Section

Description Displays a description of the error.

Error Message Displays the error message.

Error Timestamp Displays the date and time that the error occurred.

Copyright © 1988-2007, Oracle. All rights reserved.

Return When you have completed reviewing the error information, click the button to return to the previous page.

Segment Index Indicates the index of the segment inside a message.

If a message has three segments, you can look at each segment by the index. Segment index *1* is the first segment, segment index *2* is the second segment, and segment index *3* is the third segment.

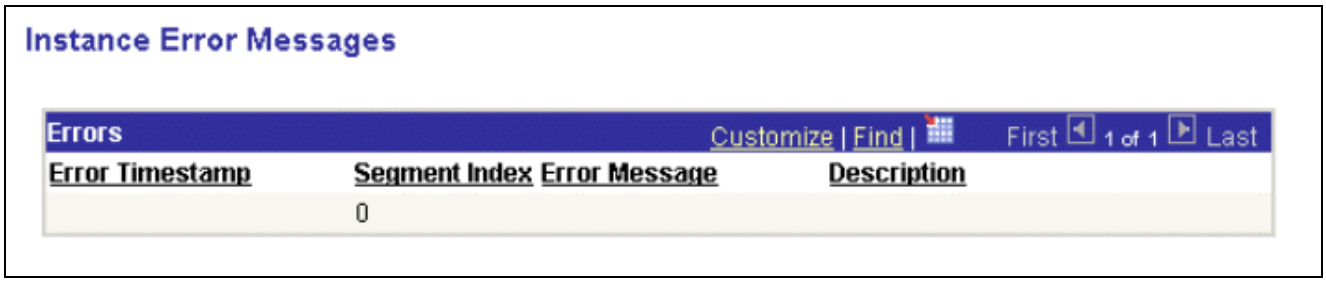
Pages Used to View Service Operation Errors

Page Name	Object Name	Navigation	Usage
Publication Contracts Error page	IB_MONITOR_ERR	On the Asynchronous Details page, in the Publication Contracts section click the Error Messages link.	View error details for publication contracts.
Subscription Contracts Error page	IB_MONITOR_ERR	On the Asynchronous Details page, in the Subscription Contracts section click the Error Messages link.	View error details for subscription contracts.
Instance Errors Messages page	IB_MONITOR_SYN_ERR	Select PeopleTools, Integration Broker, Service Operations Monitor, Monitoring, Synchronous Details. Click the Error Messages link.	View error details for synchronous service operation instances.

Viewing Asynchronous Service Operation Instance Errors

When an error occurs while processing an asynchronous service operation instance, an Error Message link appears on in the operation instance section of the Asynchronous Details page Click the link to access the Instance Error Messages page and information about the error.

The following example shows the Instance Error Messages page:



Instance Error Messages

The fields that display in this section are discussed elsewhere in this section.

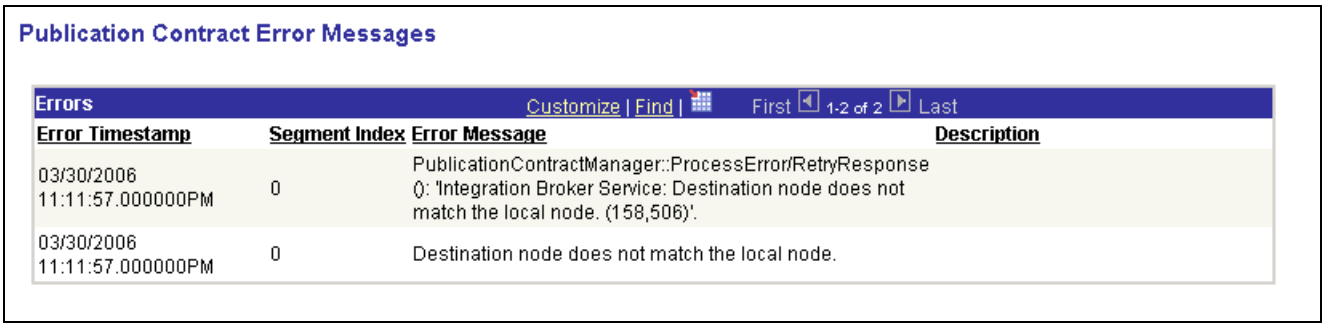
See Chapter 21, “Using the Service Operations Monitor,” Common Elements Used in This Section, page 443.

Copyright © 1988-2007, Oracle. All rights reserved.

Viewing Publication Contract Errors

When an error occurs while processing a publication contract, an Error Message link appears on the Asynchronous Details page in the Publication Contracts section on the Actions tab. Click the link to access the page and information about the error.

The following example shows the Publication Contract Error Messages page:



Publication Contract Error Messages page

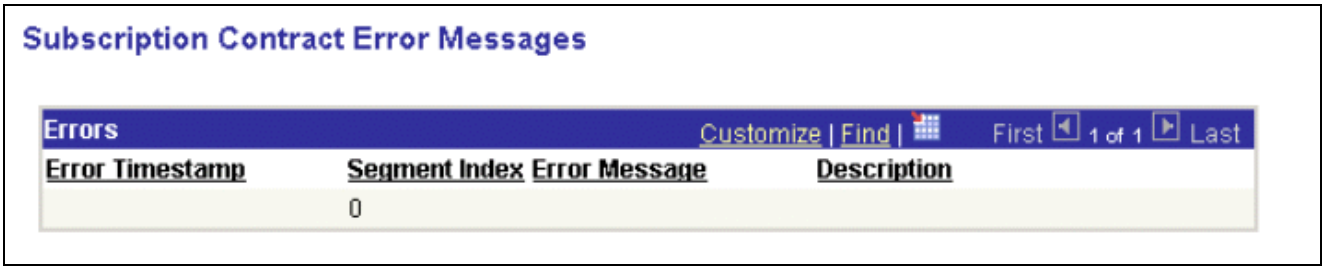
The fields that display in this section are discussed elsewhere in this section.

See Chapter 21, "Using the Service Operations Monitor," Common Elements Used in This Section, page 443.

Viewing Asynchronous Subscription Contract Errors

When an error occurs while processing a subscription contract, an Error Message link appears on the Asynchronous Details page in the Subscription Contracts section on the Actions tab. Click the link to access the page and information about the error.

The following example shows the Subscription Contract Error Messages page:



Subscription Contract Error Messages page

The page displays the following information:

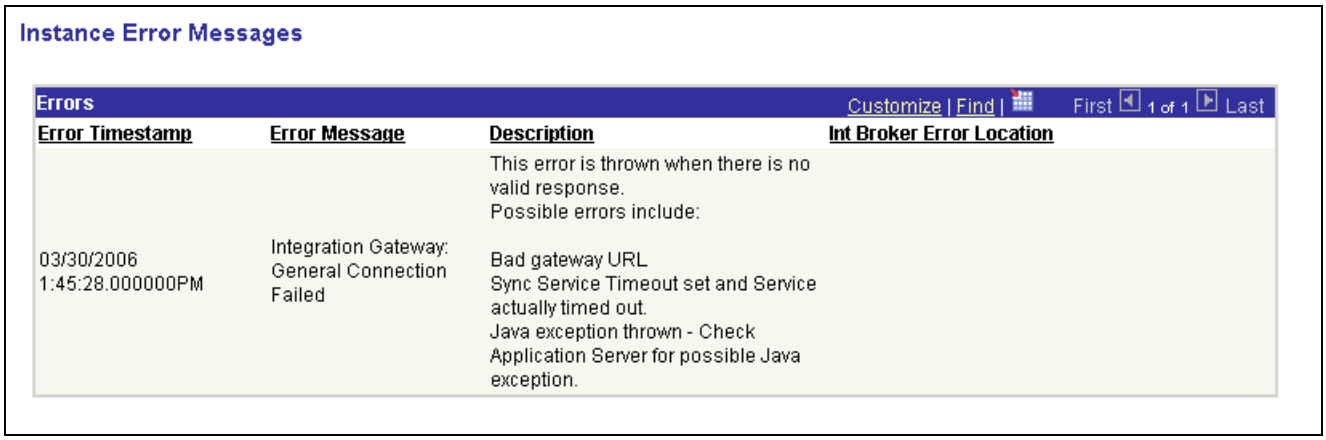
The fields that display in this section are discussed elsewhere in this section.

See Chapter 21, "Using the Service Operations Monitor," Common Elements Used in This Section, page 443.

Viewing Synchronous Service Operations Errors

When an error occurs with a synchronous service operation transaction, an Error Messages link appears on the Synchronous Details page. Click the link to access the Instance Error Messages page and details about the error.

Copyright © 1988-2007, Oracle. All rights reserved.



Instance Error Message page for synchronous service operation instance errors

The page displays the following information:

Int Broker Error Location Displays the location of the error in the PeopleSoft Integration Broker system, if known.

Other fields that display in this section are discussed elsewhere in this section.

See Chapter 21, "Using the Service Operations Monitor," Common Elements Used in This Section, page 443.

Viewing and Editing Service Operation XML

This section discusses how to:

- View service operation XML.
- Edit service operation XML.

Understanding Viewing and Editing Service Operation XML

The Service Operations Monitor enables you to view service operation XML for asynchronous service operation instances, asynchronous publication contracts, asynchronous subscription contracts and for synchronous service operation instances.

Note. You can view and edit XML only if you have the appropriate service operation permission.

Asynchronous Service Operation XML

If viewing or editing XML for a publication or subscription contract, the transformed XML appears if any transformations were applied for the publication contract or subscription contract. Use the View XML link or the Edit XML link in the service operation instance section to view and edit the original XML that was received.

Synchronous Service Operation XML

In certain situations, the XML content of a service operation isn't visible in the Service Operations Monitor. This is because of the way service operation data is logged. Initially, the log data (including the service operation XML) for any transaction is held in system memory.

Copyright © 1988-2007, Oracle. All rights reserved.

With synchronous transactions, PeopleSoft Integration Broker retains the log data in memory for a longer period, to allow for certain operations to complete. The delay before you can view the XML content in the Synchronous Details component depends on several factors, including the details of the integration and whether you're at the sending or the receiving end of the transaction. If you don't see the service operation XML content right after the service operation was transmitted, exit the Synchronous Details component and wait for a minute, then reopen the service operation and check the XML view again.

Note. For synchronous service operations, to view full service operation details in XML you must set a parameter in the routing definition for the service operation. On the Routing-Routing Definitions page, from the Log Detail dropdown list box select *Header and Detail*.

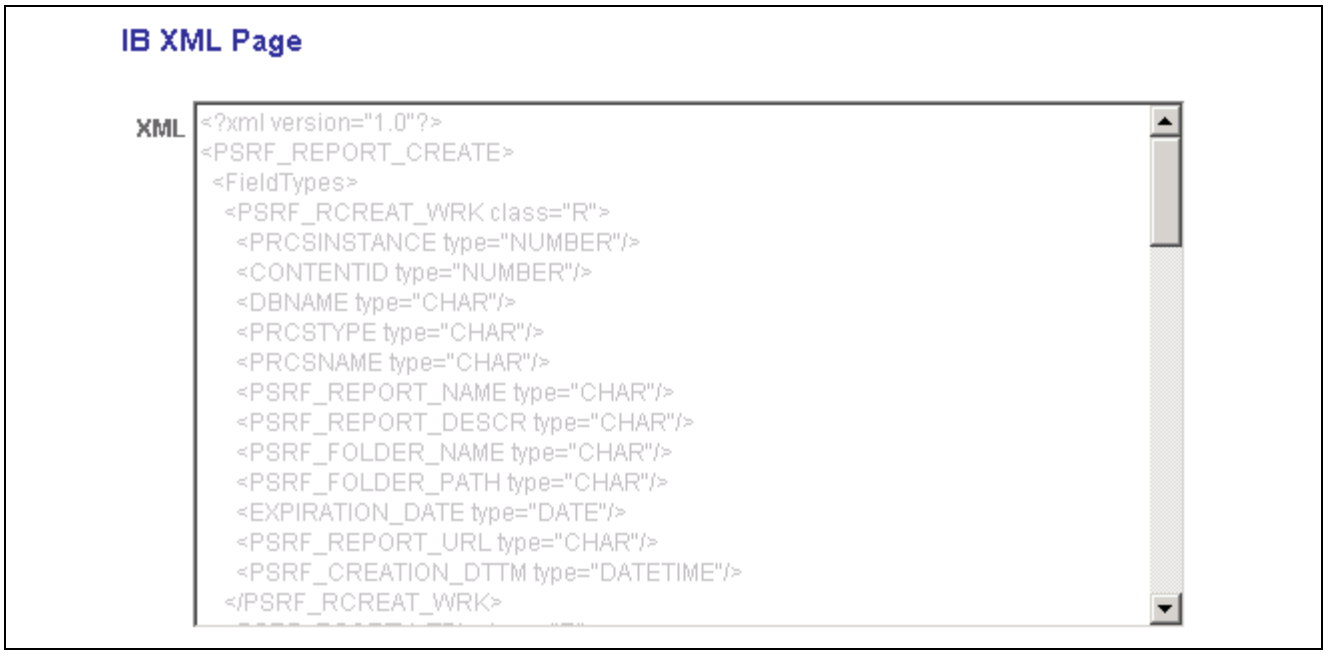
Pages Used to View and Edit Service Operation XML

Page Name	Object Name	Navigation	Usage
IB XML Page	IB_MONITOR_XML	• Select PeopleTools, Integration Broker, Service Operations Monitor, Monitoring, Asynchronous Details. Click the View XML or Edit XML link in the following locations: Under service operation details to view or edit service operation XML; In the Publication Contracts section to view or edit XML for a publication contract; and in the Subscription Contracts section to view or edit XML for a subscription contract. • Select PeopleTools, Integration Broker, Service Operations, Monitor, Synchronous Details. Click the View XML or Edit XML link to view or edit synchronous service operation XML.	Use the IB XML page to view or edit service operation XML. The View XML link appears when there are no errors with a transaction and provides read-only access to the service operation XML. The Edit XML link appears when there are errors with the transaction. Click the link to edit the service operation XML to correct errors.

Viewing Service Operation XML

You can view the XML for an asynchronous service operation instance, publication contract, subscription contract or synchronous service operation by clicking the View XML link. When you click the link, the IB XML page appears and displays the data in read-only format, as shown in the following example:

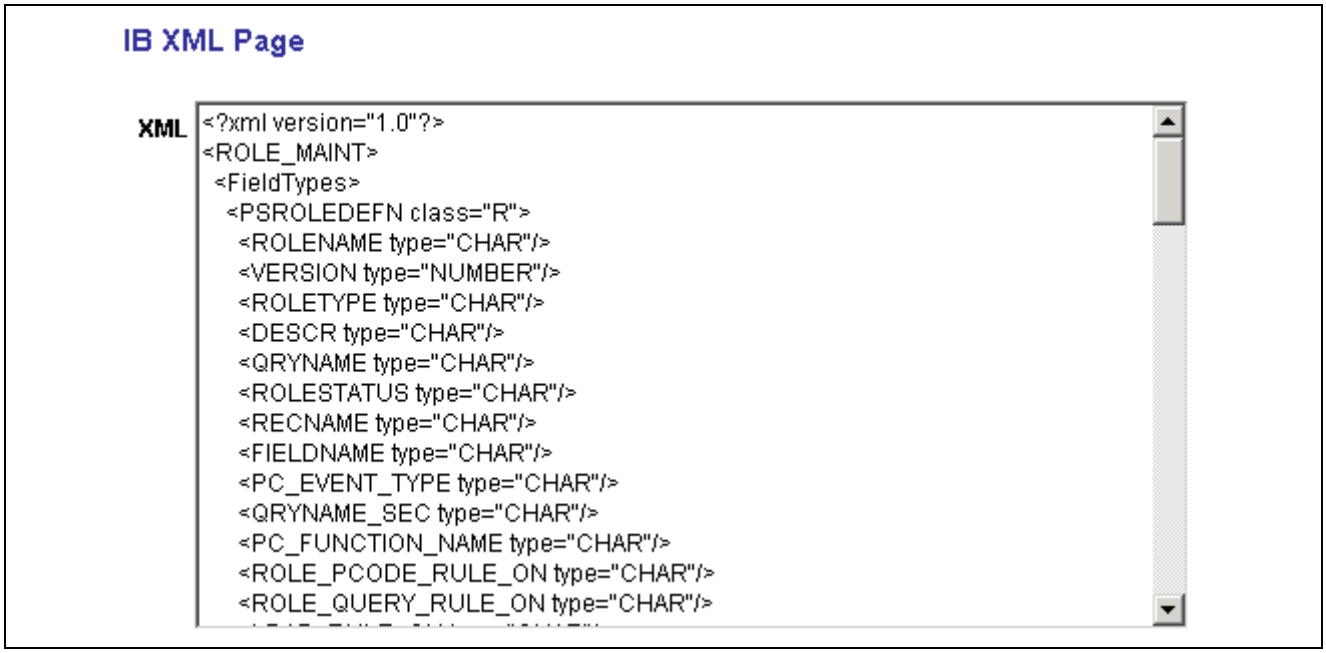
Copyright © 1988-2007, Oracle. All rights reserved.



IB XML Page displaying XML in read-only format

Editing Service Operation XML

When an error occurs processing an asynchronous service operation instance, publication contract subscription contract, or synchronous service operation instance, an Edit XML link appears, as shown in the following example:



IB XML page

The page enables you to edit the XML to correct any errors.

To edit XML you must have the appropriate permissions to the service operation and the service operation must have a status of *New, Error, Retry, Timeout, Edited* or *Cancelled.*

Copyright © 1988-2007, Oracle. All rights reserved.

When you have completed editing the XML click the Save button to save your changes. Click the Return button to return to the Asynchronous Details page.

Viewing Service Operation Nonrepudiation Signature Information

This section discusses how to view nonrepudiation signature information in XML format for service operations.

Understanding Viewing Service Operation Nonrepudiation Signature Information

If an asynchronous or synchronous service operation is sent with a signature you can view the nonrepudiation signature in XML format.

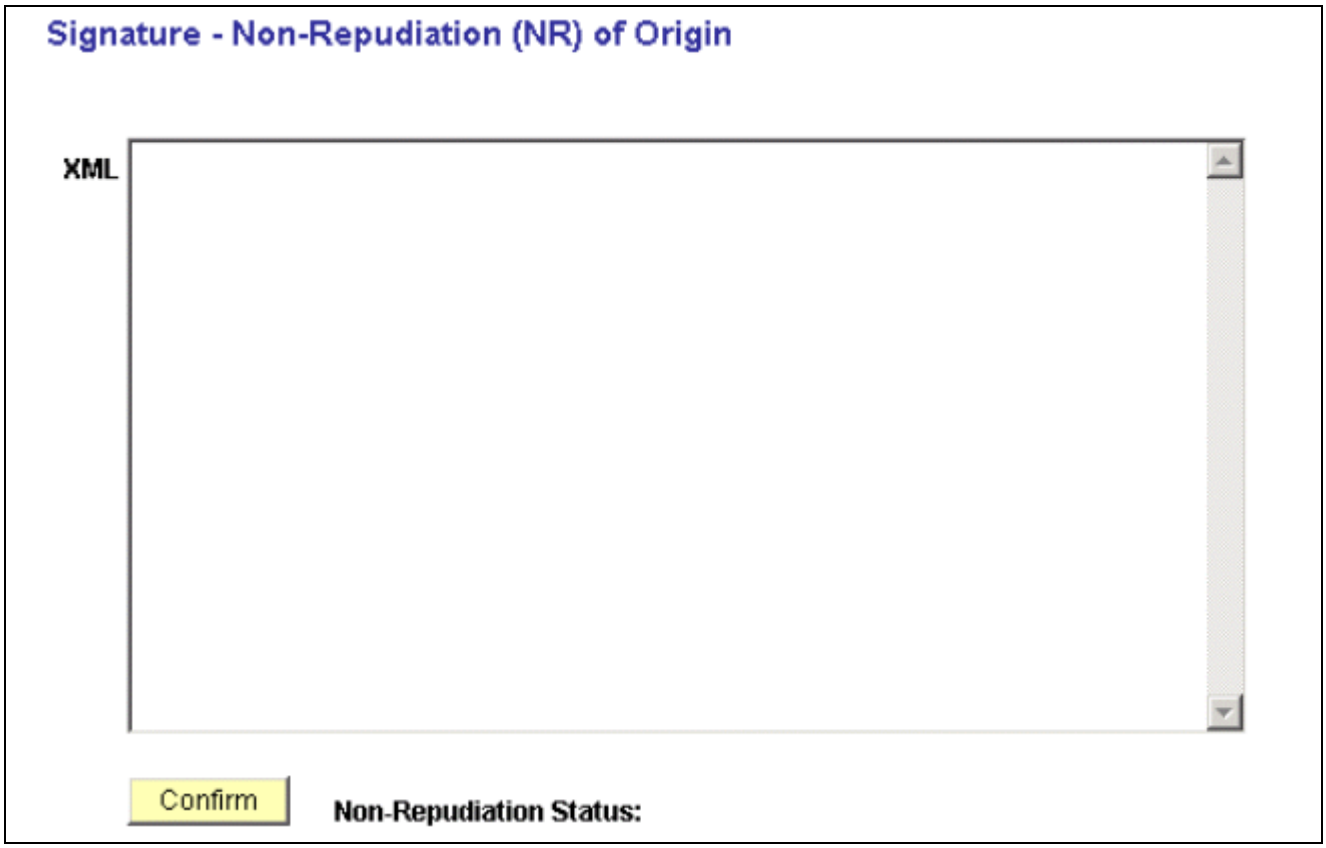
Pages Used to View Service Operation Nonrepudiation Signature Information

Page Name	Object Name	Navigation	Usage
Signature-Non-Repudiation (NR) of Origin page	IB_MONITOR_NRXML	Use one of the following methods to access the page: • Select PeopleTools, Integration Broker, Service Operations Monitor, Monitoring, Asynchronous Details. In the Publication Contracts section, click the Information tab. Click the Signature link. • Select PeopleTools, Integration Broker, Service Operations Monitor, Monitoring, Synchronous Details. Click the Signature link next to the Non-Repudiation ID field.	View the nonrepudiation signature information in XML format.

Viewing Nonrepudiation Signatures in XML Format

The Signature (NR) page displays nonrepudiation information for asynchronous service operations. The following example shows the page:

Copyright © 1988-2007, Oracle. All rights reserved.



Signature-Non-Repudiation (NR) of Origin page

The Signature link to this page appears only if the service operation is sent with a signature. When you click the Signature link, the service operation signature appears in XML.

Click the Confirm button to confirm the nonrepudiation status. Click the Return button to return to the previous page.

Running Batch Error Notification Processes

This section provides an overview of batch error notification and discusses how to run batch error notification programs.

Understanding Batch Error Notification

Although you can easily use the Service Operations Monitor to scan your system for service operations, that approach requires you to launch the Service Operations Monitor on a scheduled basis to search for any issues affecting the messaging system. The Error Notification component (PT_ERR_RUNCNTL) provides access to an Application Engine batch program, PT_AMM_WF, that you can schedule to run on a recurring basis.

To access the program, select PeopleTools, Integration Broker, Monitor Integrations, Error Notification.

Note. You can use PT_AMM_WF to notify users of errors relating to asynchronous service operations only.

 Copyright © 1988-2007, Oracle. All rights reserved.

The following table describes the information for which PT_AMM_WF scans, how it notifies administrators, and what administrators should do after receiving an error notification.

Step	Task	Description
1	Query Message Queues	The program scans the following messaging queues in the database in search of service operation with a status of either *Error* or *Timeout*. • Publications Contracts Queue • Subscriptions Contracts Queue
2	Trigger Workflow	Upon encountering a service operation status of either *Error* or *Timeout*, PT_AMM_WF sends a workflow to all users assigned to the APP_MSG_ADMINISTRATOR role at runtime. The query for this role associates a service operation with a user through the service operation's queue name property. All users that have at least read-access to the service operation queue are notified.
3	Resolve Issue	Administrators also receive a new worklist item reflecting the problematic service operation. To access the service operation, an administrator clicks the item in the worklist. The link leads to the Asynchronous Details component. The component is presented with the specified service operation loaded.

Prerequisites for Using Batch Error Notification

To enable the workflow notification functionality, you need to have the following items in place within security definitions:

- Grant access to the PT_AMM_DUMMY component interface. Navigate to PeopleTools, Security, Permissions & Roles, Permission Lists, Component Interfaces.
- Assign users to the APP_MSG_ADMINISTRATOR role using PeopleTools, Security, Permissions & Roles, Roles, Members.

  PeopleSoft delivers the APP_MSG_ADMINISTRATOR role.
- Add email addresses for users assigned to the APP_MSG_ADMINISTRATOR role to their user profiles so that they can receive the notification.

  To complete this task, select PeopleTools, Security, User Profiles, User Profiles, Edit Email Addresses

See Also

Enterprise PeopleTools 8.49 PeopleBook: Security Administration

Creating Static Error Notification Lists

By default, error notifications are sent to all users who can monitor the service operation queue; these are users assigned to the Query role. However, you can send error notifications to a static list of users that belong to the APP_MSG_ADMINISTRATOR role. To do so you must turn off the Use Query to Route Workflow option for the APP_MSG_ADMINISTRATOR role, and assign specific users to the role.

Copyright © 1988-2007, Oracle. All rights reserved.

To view users assigned to the APP_MSGADMINISTRATOR role, run the _ROLE_APP_MSG_ ADMINISTRATOR query.

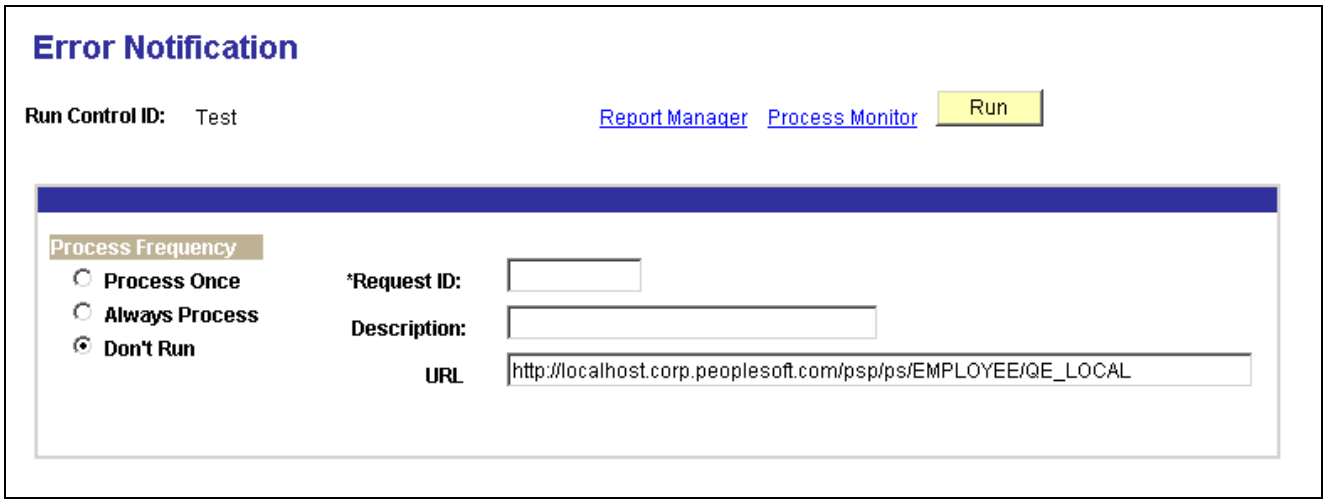
To create a static error notification list:

1. Turn off the User Query to Route Workflow option.
    a. Select PeopleTools, Security, Permissions & Roles, Roles.
    b. Select and open the APP_MSG_ADMINISTRATOR role.
    c. Click the Workflow tab.
    d. In the Workflow Routing Options box, clear the Use Query to Route Workflow option, and click Save.
2. Assign specific users to the APP_MSG_ADMINISTRATOR role.

    See *Enterprise PeopleTools 8.49 PeopleBook: Security Administration*, "Administering User Profiles," Specifying User Profile Attributes.

Running Batch Error Notification

You use the Error Notification page to run the PT_AMM_WF process. To access the page, select PeopleTools, Integration Broker, Monitor Integrations, Error Notification.



Error Notification page

To run PT_AMM_WF:

1. Select PeopleTools, Integration Broker, Monitor Integrations, Error Notification.
2. Select an existing run control ID, or add a new one using the Add button.

    The Error Notification page appears.
3. Select a process frequency.

    Options are:

    - *Process Once*. Select to run PT_AMM_WF manually.
    - *Process Always*. Select to run PT_AMM_WF constantly.
    - *Don't Run*. Select to disable a recurring PT_AMM_WF run.
4. Add a request ID and description.

Copyright © 1988-2007, Oracle. All rights reserved.

These attributes uniquely identify a run control. You only see the IDs when you have a list of run controls.

5. In the URL field, enter the PeopleSoft Pure Internet Architecture URL to provide in the email error notification. Users use the URL to link to the error.

   The URL of the current web server displays in this field by default.

6. ClickRun.
7. Click OK on the Process Scheduler Request page to submit the process.

Archiving Service Operation Instances

This section discusses how to:

- Archive service operation instances.
- Retrieve archived service operation instances.

Prerequisites

To archive service operation instances you must activate archiving on the service operation queue.

See Chapter 11, “Managing Service Operation Queues,” page 197.

Copyright © 1988-2007, Oracle. All rights reserved.

Pages Used to Archive Service Operation Instances

Page Name	Object Name	Navigation	Usage
Monitor Overview page	IB_MONITOR_OVRVIEW	Select PeopleTools, Integration Broker, Service Operations Monitor, Monitoring, Asynchronous Services. Click the Monitor Overview tab.	Use this page to archive asynchronous service operations.
Operation Instances page	IB_MONITOR_PUBHDR	Select PeopleTools, Integration Broker, Service Operations Monitor, Monitoring, Asynchronous Services. Click the Operation Instance tab.	Use this page to archive asynchronous service operation instances.
Publication Contracts page	IB_MONITOR_PUBCON	Select PeopleTools, Integration Broker, Service Operations Monitor, Monitoring, Asynchronous Services. Click the Publication Contracts tab.	Use this page to archive publication contracts.
Subscription Contracts page	IB_MONITOR_SUBCON	Select PeopleTools, Integration Broker, Service Operations Monitor, Monitoring, Asynchronous Services. Click the Subscription Contracts tab.	Use this page to archive subscription contracts.
Synchronous Services page	AMM_SYNCMSGLIST	Select PeopleTools, Integration Broker, Service Operations Monitor, Monitoring, Synchronous Services.	Use this page to archive synchronous service operations.

Archiving Service Operations

You can archive service operation instances one at a time from the Asynchronous Details component or the Synchronous Details component by clicking the Archive button that appears on the right side of the page.

See Also

Chapter 21, “Using the Service Operations Monitor,” Viewing Asynchronous Service Operation Instance Details, page 431

Chapter 21, “Using the Service Operations Monitor,” Viewing Synchronous Service Operation Details, page 438

Retrieving Archived Messages

You can retrieve archived service operation instances from the following pages in the Asynchronous Services component: Monitor Overview page, Operation Instances page, Publication Contracts page, and Subscription Contracts page.

 Copyright © 1988-2007, Oracle. All rights reserved.

To retrieve archived service operations instances, select theArchive check box and click Refresh. Archived service operations appear in the results grid on the page. For any returned row, click the Details link to view the service operation header and service operation content.

Running Batch Service Operation Archiving Processes

This section discusses how to run batch service operation archiving processes.

Understanding Running Batch Service Operation Archiving Processes

For performance and general maintenance reasons, you may want to archive older service operation to clear space on your live runtime monitor tables.

The PeopleSoft system provides an Application Engine program that scans all of the runtime monitor tables in the system for service operation archiving purposes. You use the Archive Messages component (RUN_APMSGARCH) to access the program.

You can use the Run Archive page to archive all service operations with a status of *Done* or *Cancel*. Or you can archive service operations based on their status, their age, or a combination of the two. For example, you can choose to archive service operations with a status of *Done* that have been in the messaging system for more than 14 days.

Prerequisites for Running Batch Service Operation Archiving Processes

Before you run a batch service operation archive process, inactivate the pub/sub server domain. Then, after you run the process, reactive the pub/sub server domain.

See Also

Chapter 5, “Using the Integration Broker Quick Configuration Page,” page 37

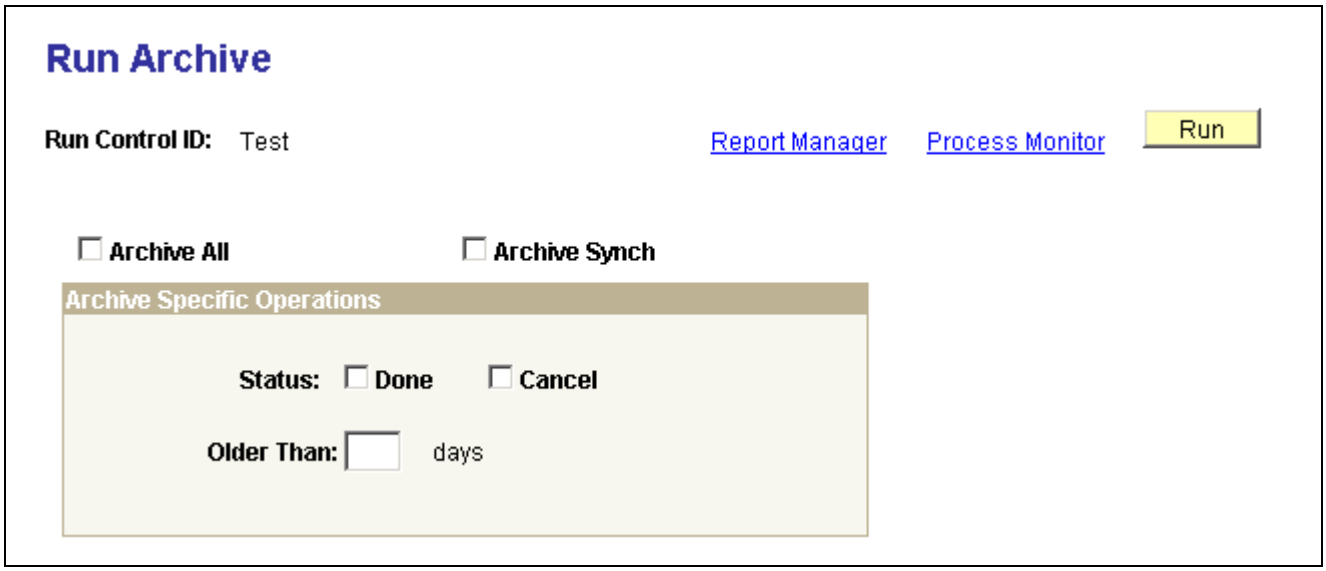
Page Used to Run Batch Service Operation Archiving Processes

Page Name	Object Name	Navigation	Usage
Run Archive page	RUN_APMSGARCH	Select PeopleTools, Integration Broker, Service Operations Monitor, Monitoring, Archive Monitor Data.	Use this page to archive service operations in batch.

Running Batch Service Operation Archiving Processes

You use the Run Archive page to invoke the archive process. To access the page, select PeopleTools, Integration Broker, Service Operations Monitor, Monitoring, Archive Monitor Data. The following example shows the Run Archive page:

Copyright © 1988-2007, Oracle. All rights reserved.



Run Archive page

To run the batch service operation archiving processes:

1. Select PeopleTools, Integration Broker, Service Operations Monitor, Monitoring, Archive Monitor Data.
2. Select an existing run control ID, or add a new one.

   The Run Archive page appears.
3. Select the service operation to archive

   Archive All — Select this check box to archive all service operations in the messaging system with a status of *Done* or *Cancel*, regardless of how long they have been in the messaging system.

   Archive Synch — Check the box to archive all synchronous service operations.

   Warning! Leaving the box unchecked deletes all synchronous service operations.

   Done — Select to archive service operations in the messaging system with a status of *Done*.

   Cancel — Select to archive only those service operations in the messaging system with a status of *Cancel*.

   Older Than — Enter a numeric value between 1 and 999. Do not enter 0 or a decimal value. Messages older, in days, than the value that you enter will be archived.

   This option archives service operations based on days. If the date is August 15, 2004, service operations dated August 13, 2004 and earlier are archived.
4. Click the Run button.

   The Process Schedule Request page appears.
5. Make the appropriate selections, and click OK.

Copyright © 1988-2007, Oracle. All rights reserved.

Note. Using APPMSGARCH to archive service operation data is the batch approach. You can also archive individual service operations online using the Archive option on the Asynchronous Services-Monitor Overview page and the Synchronous Services page.

See Also

Chapter 21, "Using the Service Operations Monitor," Monitoring Service Operation Transactions, page 422

Chapter 21, "Using the Service Operations Monitor," Monitoring Synchronous Service Operations, page 435

Enterprise PeopleTools 8.49 PeopleBook: PeopleSoft Process Scheduler, "Using Process Monitor"

Enterprise PeopleTools 8.49 PeopleBook: PeopleSoft Process Scheduler, "Using Report Manager"

Viewing System Performance Statistics

This section provides an overview of messaging system performance statistics and discusses how to:

- Enable the messaging system performance statistics feature.
- Filter statistics data.
- View inbound asynchronous post statistics.
- View broker handler statistics.
- View subscription contract handler statistics.
- View publication contract handler statistics.
- View inbound synchronous service operations statistics.
- View outbound synchronous service operations statistics.
- View local synchronous service operations statistics.

Understanding Messaging System Performance Statistics

The Service Operations Monitor provides a Statistics page that enables you to view performance statistics for asynchronous and synchronous service operations that flow through PeopleSoft Integration Broker.

The Service Operations Monitor provides performance statistics for the flow of asynchronous service operations through the pub/sub system. Statistics are captured for the flow of service operations through the following pub/sub components:

Asynchronous Pub/Sub Component	Description
Inbound Asynchronous Post.	Includes posting service operations to the integration gateway, as well as posting service operations to the application server and putting them into queue. Also includes processing time for creating and sending response messages.

Copyright © 1988-2007, Oracle. All rights reserved.

Asynchronous Pub/Sub Component	Description
Broker Handler.	Analyzes all service operations in the queue and determines the transaction type. Based on the transaction type, creates a subscription contract, publication contract, or both.
Subscription Contract Handler.	Runs PeopleCode associated with the service operation.
Publication Contract Handler.	Routes the service operation to another destination.

In addition, the Service Operations Monitor provides performance statistics for the following synchronous transaction types.

Synchronous Transaction Type	Description
Inbound Synchronous.	Inbound service operations from PeopleSoft and non-PeopleSoft systems. The sending system requires a response from the receiving PeopleSoft system before additional processing occurs.
Outbound Synchronous.	Outbound service operations from PeopleSoft systems. The sending system requires a response from the receiving system (PeopleSoft or non-PeopleSoft) before additional processing occurs
Local Synchronous.	A local PeopleSoft system that sends inbound or outbound service operations to itself. This scenario is typically used for testing purposes. However this scenario can also apply to two PeopleSoft systems communicating behind the same firewall.

In the case of asynchronous service operation flow through the publication contract handler and outbound synchronous transactions, you can capture statistics about performance on remote PeopleSoft systems.

Common Elements Used in this Section

Click to remove section tabs and show all columns of information.

Deselect All Click to deselect all nodes.

External Service Name Indicates the name of the service operation sent by the sending node.

Publish Node, Publishing Node Name of the publishing node. The local default node is the default.

Purge All Asynchronous Timings, Purge All Synchronous Timings, Purge All Broker Timings, and **Purge All Publication Timings** Click one of these buttons to purge performance statistics for the given service operation when you are done using the statistics or want to view new performance statistics.

Queue and**Queue Name** Name of the queue to which a service operation or service operations belong.

Copyright © 1988-2007, Oracle. All rights reserved.

Refresh Click the Refresh button to refresh the contents of the page with the search criteria selected.

Select Check the box to include statistics for the node when viewing the data in bar chart or pie chart format.

Service Operation Select the name of the service operation for which to view performance statistics.

This field displays when working with subscription contract handler, publication contract handler, inbound synchronous, outbound synchronous and local synchronous transactions.

Subscription Name Displays the identifier for PeopleCode that is associated with the service operation.

Subscriber Node Displays the name of the subscribing node.

Timestamp Displays the date and time that the service operation flowed through the system.

Time Period The Time Period group box features four fields for searching by date and time: From Date, To Date, From Time and To Time.

All must be left blank or all must be populated. When left blank, no date or time is used as part of the search criteria.

Transaction ID The unique identifier for the transaction assigned by the system.

View Bar Chart Click to view performance statistics in bar chart format. Values shown are averages.

View Pie Chart Click to view performance statistics in pie chart format. Values shown are averages.

Pages Used to View System Performance Statistics

Page Name	Object Name	Navigation	Usage
Monitor Setup Options	IB_MONITORADMIN	Select PeopleTools, Integration Broker, Service Operations Monitor, Administration, Monitor Setup Options.	Enable the performance statistics feature on the application server.
Gateway Properties page	IBGWPROPERTIESPAGE	Select PeopleTools, Integration Broker, Integration Setup, Gateways. The Gateways page appears. Click the Gateway Setup Properties link. Enter the gateway properties user ID and password and click OK. The PeopleSoft Node Configuration page appears. Click the Advanced Properties link.	Enable the performance statistics feature on the integration gateway.

Copyright © 1988-2007, Oracle. All rights reserved.

Page Name	Object Name	Navigation	Usage
Inbound Asynchronous Post page	AMM_ASYN_STATS	Select PeopleTools, Integration Broker, Service Operations Monitor, Monitoring, Statistics. Click the Inbound Asynchronous Post link.	View statistics for inbound asynchronous service operations.
Broker Handler page	AMM_BRK_STATS	Select PeopleTools, Integration Broker, Service Operations Monitor, Monitoring, Statistics. The Statistics page appears. Click the Broker Handler link.	View system performance statistics for asynchronous service operations instances.
Subscription Contract Handler page	AMM_SUB_STATS	Select PeopleTools, Integration Broker, Service Operations Monitor, Monitoring, Statistics. The Statistics page appears. Click the Subscription Contract Handler link.	View system performance statistics for asynchronous service operation subscriptions.
Publication Contract Handler page	AMM_PUB_STATS	Select PeopleTools, Integration Broker, Service Operations Monitor, Monitoring, Statistics. The Statistics page appears. Click the Publication Contract Handler link.	View system performance statistics for asynchronous service operation publications.
Inbound Synchronous page	AMM_INSYNC_STATS	Select PeopleTools, Integration Broker, Service Operations Monitor, Monitoring, Statistics. The Statistics page appears. Click the Inbound Synchronous link.	View system performance statistics for inbound synchronous service operations.
Outbound Synchronous	AMM_SYNC_STATS	Select PeopleTools, Integration Broker, Service Operations Monitor, Monitoring, Statistics. The Statistics page appears. Click the Outbound Synchronous link.	View system performance statistics for outbound synchronous service operations.
Local Synchronous page	AMM_SYNC_STATS	Select PeopleTools, Integration Broker, Service Operations Monitor, Monitoring, Statistics. The Statistics page appears. Click the Local Synchronous link.	View system performance statistics for local–to–local synchronous service operations.

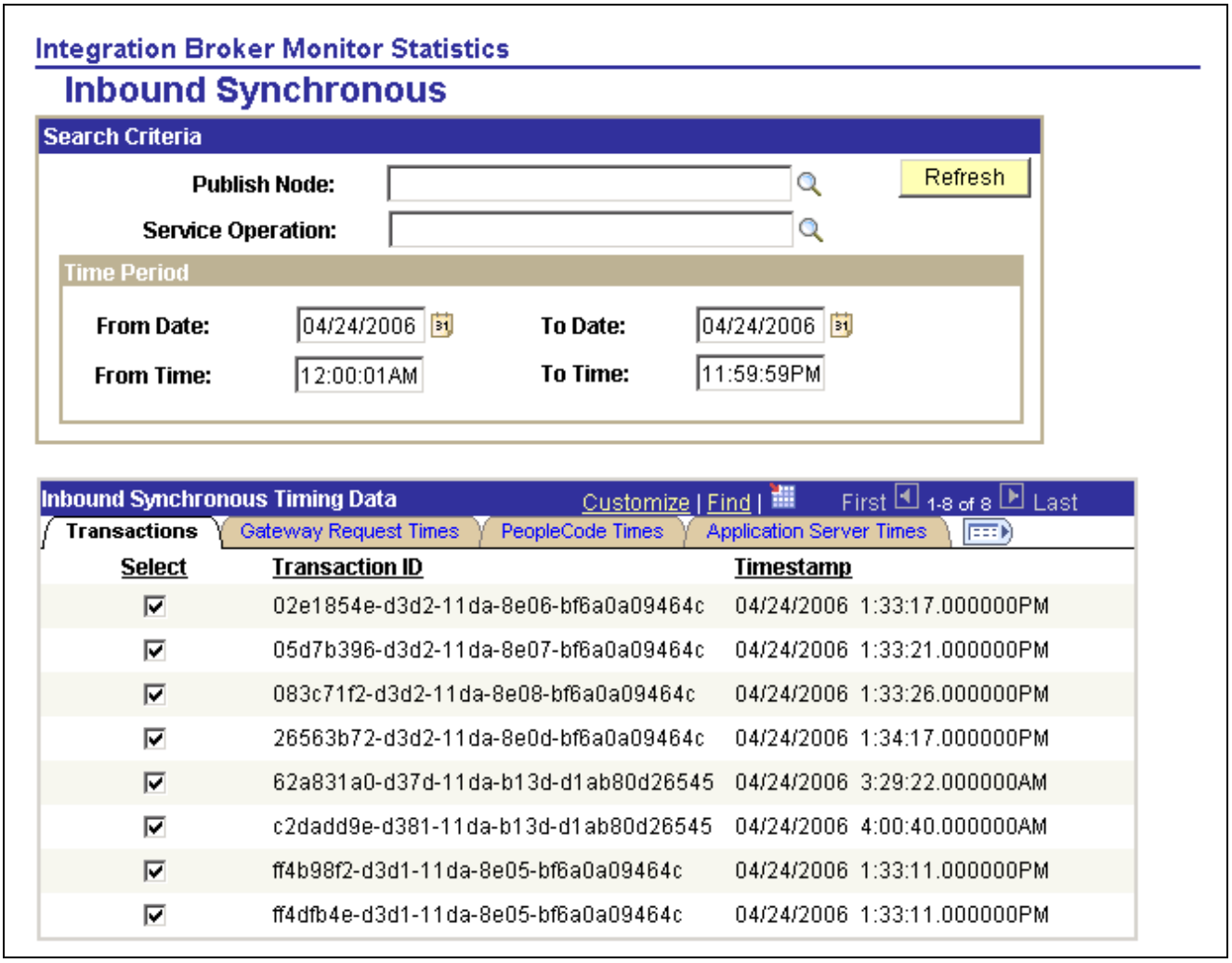
Viewing Messaging System Performance Statistics

You can view messaging system performance statistics in numeric or graphical format.

By default, the statistics appear in numeric format on tabbed pages.

Copyright © 1988-2007, Oracle. All rights reserved.

Note. All times shown on the Statistics pages are in milliseconds.



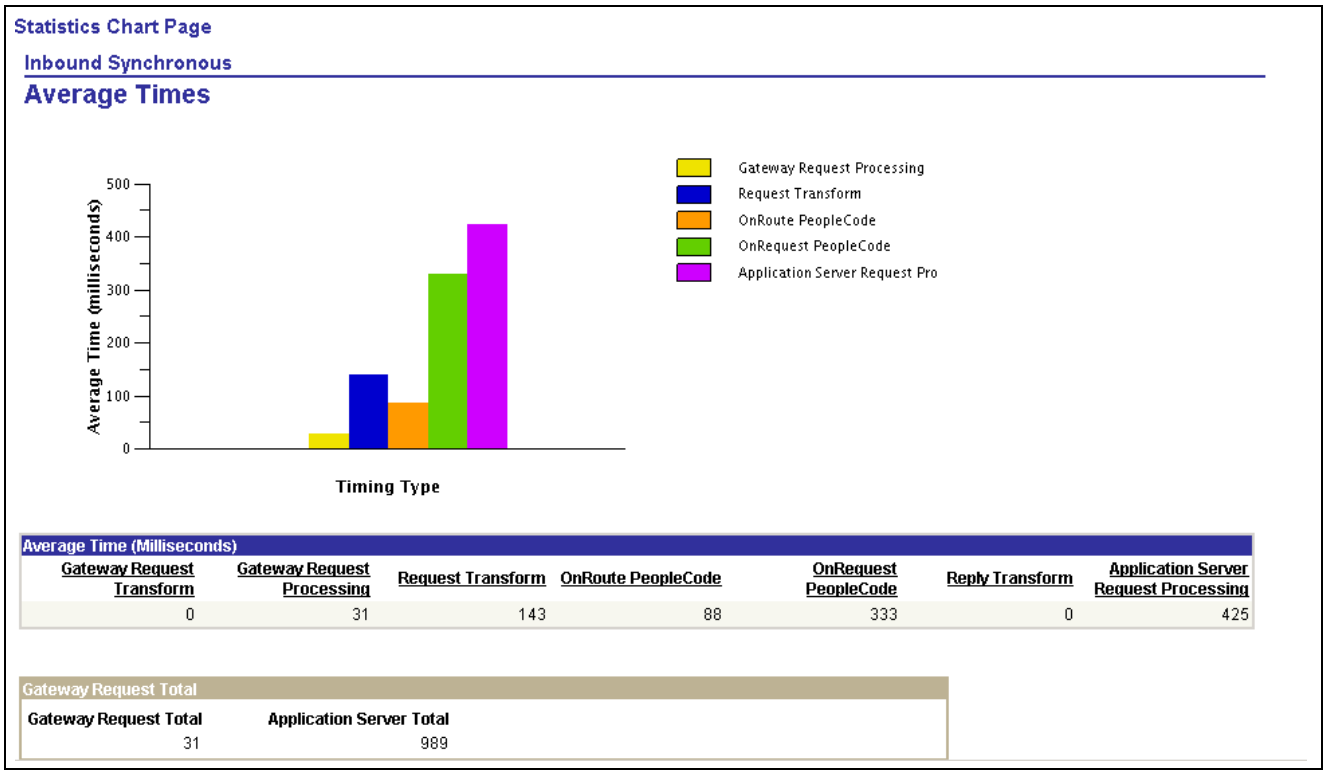
Inbound synchronous performance statistics in numeric format

You can select to view the statistics in graphical bar chart or pie chart format from any of the pages by using the View Bar Chart and View Pie Chart buttons. When you click either button, another page opens and displays the statistics in the graphical format. To return to the numeric format view of the data, click the Return button at the bottom of the page.

Note. Data displayed in bar chart and pie chart formats are averages.

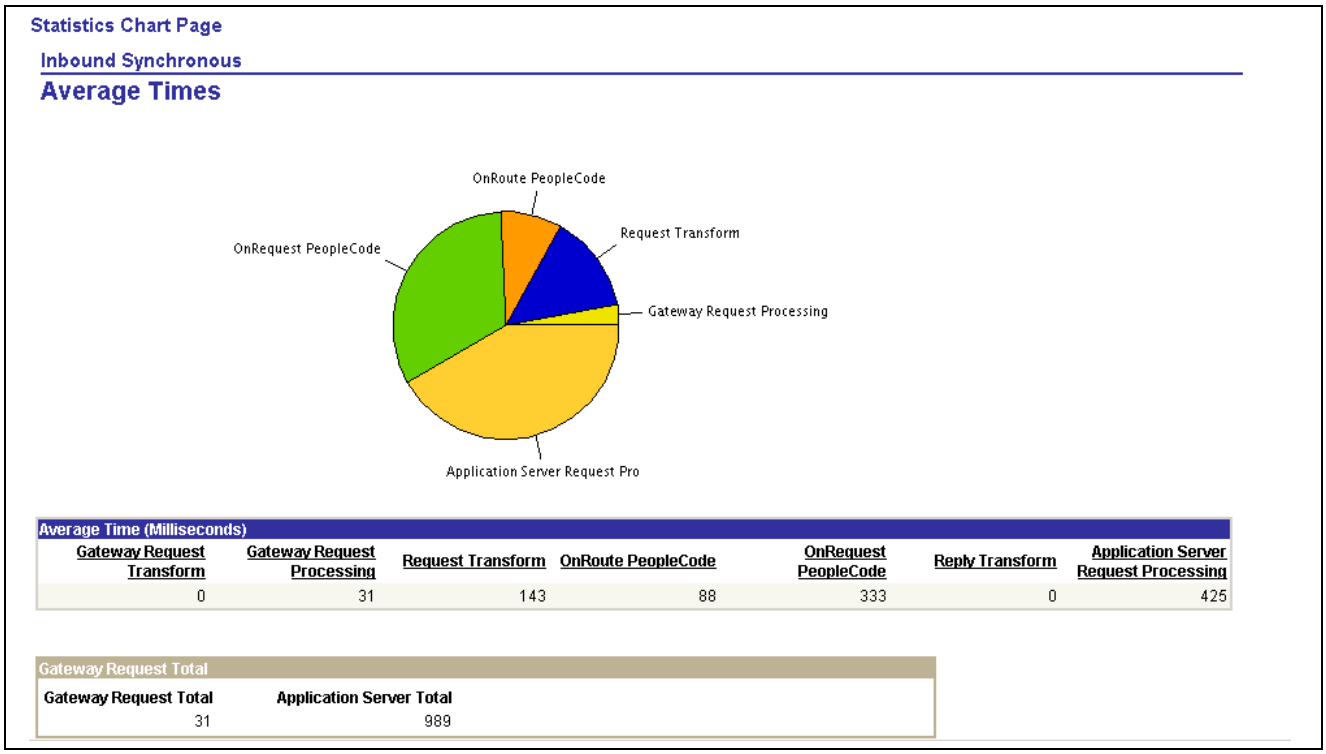
The following example shows the inbound synchronous performance statistics in bar chart format:

Copyright © 1988-2007, Oracle. All rights reserved.



Inbound synchronous performance statistics in bar chart format

The following example shows the same statistics in pie chart format:



Inbound synchronous performance statistics in pie chart format

Copyright © 1988-2007, Oracle. All rights reserved.

Enabling the Performance Statistics Feature

To view messaging system performance information, you must enable the statistics feature on the Monitor Setup Options page in the Service Operations Monitor as well as on integration gateway through the integrationGateway.properties file.

The setting in the Service Operations Monitor enables the system to capture performance statistics for activity on the application server. The setting on the integration gateway enables the system to capture performance statistics for activity on the integration gateway.

If you enable the feature only in the Service Operations Monitor and not on the integration gateway, the system captures statistics only for activity on the application server and does not capture any information for activity on the integration gateway.

Note. It is recommended that you enable the statistics feature on both the application server and on the integration gateway.

You do not need to perform any setup tasks in the integration gateway or in the Service Operations Monitor to capture performance statistics on the remote PeopleSoft system. The sending PeopleSoft system includes an identifier in the message request that prompts the remote PeopleSoft system for performance information. This information is returned as part of the response message.

To enable the statistics feature on the application server:

1. Select PeopleTools, Integration Broker, Service Operations Monitor, Administration, Monitor Setup Options.

   The Monitor Setup Options page appears.
2. Check the IB Profile Status On check box.
3. Click the Save button.

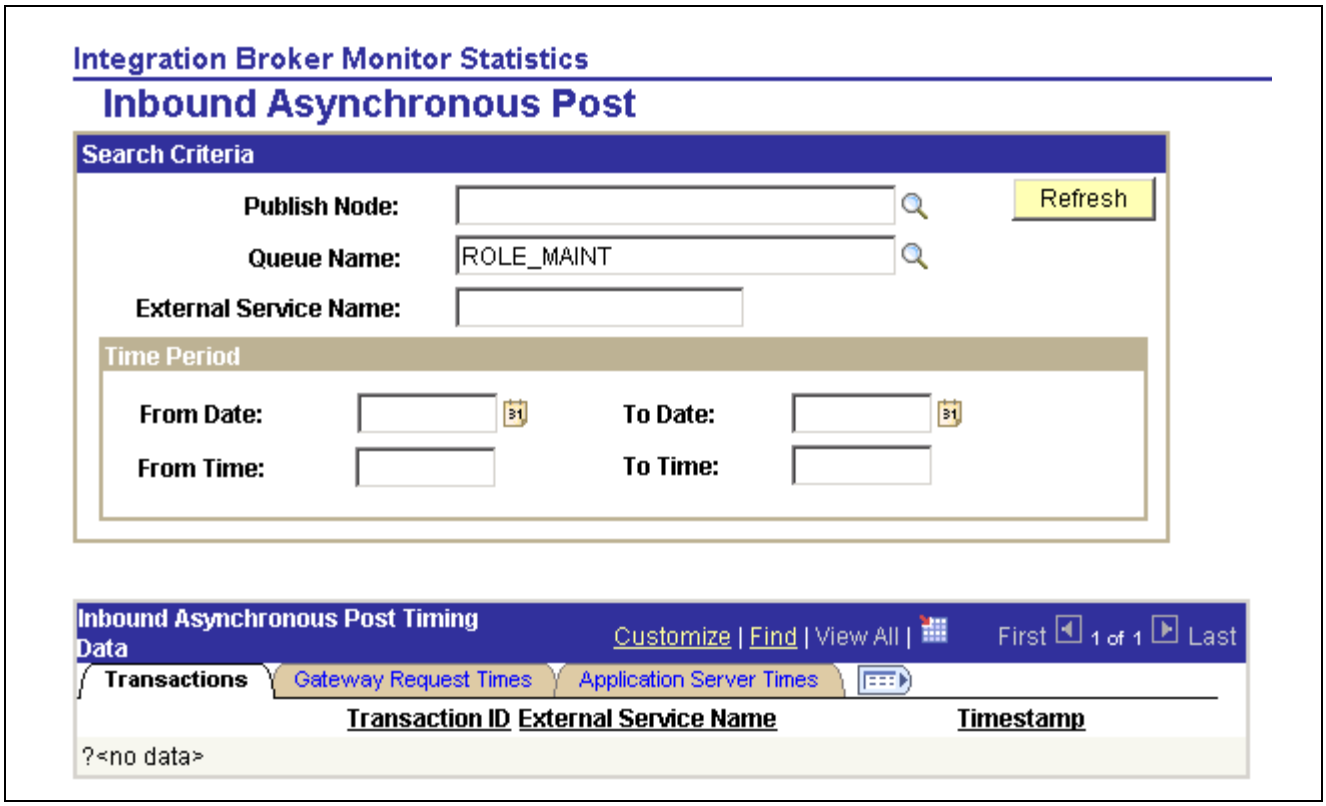
To enable the statistics feature on the integration gateway:

1. Access the integrationGateway.properties file.
2. Locate the Profile Information section at the end of the file.
3. Set the ig.ProfileInformation property to *TRUE*.
4. Save the file and refresh the integration gateway.

Selecting Statistics Data to View

When you select an transaction type for which to view statistics, you can filter data by various criteria.

Copyright © 1988-2007, Oracle. All rights reserved.



Search Criteria dialog box for inbound asynchronous transactions

The following common elements are used to filter statistics data.

To view select statistics data to view:

1. Select PeopleTools, Integration Broker, Service Operations Monitor, Monitoring, Statistics. The Statistic page displays.
2. Click the link that corresponds to the transaction type for which you want to view statistics. The choices are:
   - *Inbound Asynchronous Post.*
   - *Broker Handler.*
   - *Subscription Contract Handler.*
   - *Publication Contract Handler.*
   - *Inbound Synchronous.*
   - *Outbound Synchronous.*
   - *Local Synchronous.*

   The Search Criteria dialog box for the transaction type you selected displays.
3. Enter values in the Search Criteria dialog box, based on the data you want to view.

   See Chapter 21, "Using the Service Operations Monitor," Common Elements Used in this Section, page 458.
4. Click the Refresh button.

 Copyright © 1988-2007, Oracle. All rights reserved.

Viewing Inbound Asynchronous Post Statistics

The following table describes the fields and data that displays on the Transactions, Gateway Request Times and Application Server Times tabs on the Inbound Asynchronous Post page. This information also applies to the information that appears when you view the data in graphical format.

Tab	Field	Description and Comments
Transactions	All fields	See Common Elements Used in This Section. See Chapter 21, "Using the Service Operations Monitor," Common Elements Used in this Section, page 458.
Gateway Request Times	Gateway Request Transform	Indicates the amount of time that the system processed the gateway request transform.
Gateway Request Times	Gateway Request Processing	Indicates the amount of time that the system performed miscellaneous activities on the integration gateway, including: • Determination of which connector hands off the service operation. • Data deserialization and serialization. • Jolt request to the application server.
Gateway Request Times	Gateway Request Total	Indicates the total amount of time that the system performed processing on the integration gateway. This value is the sum of the following fields: • Gateway Request Transform. • Gateway Request Processing times.
Application Server Times	Application Server	Indicates the amount of time that processing took place on the application server. Includes: • Integration service authentication (node password, nonrepudiation, and so on). • Data deserialization and serialization. • Data decompression.

Copyright © 1988-2007, Oracle. All rights reserved.

Tab	Field	Description and Comments
Application Server Times	IB Processing Time	Indicates the time spent processing the request. This value does not include database insert processing time, PeopleCode processing time, and so forth.
Application Server Times	Integration Service Total	Indicates the total amount of time that the system performed processing on the application server. This value is the sum of the following fields: • Application Server. • IB Processing Time

Viewing Broker Handler Statistics

The following table describes the fields and data that appear on the Transactions and Times tabs on the Broker Handler page. This information also applies to the information that appears when you view the data in graphical format.

Tab	Field	Description and Comments
Transactions	All fields	See Common Elements Used in This Section. See Chapter 21, "Using the Service Operations Monitor," Common Elements Used in this Section, page 458.
Times	OnRoute PeopleCode	Indicates the amount of time that the systems took to process OnRoute PeopleCode.
Times	Inbound Transform	Indicates the amount of time that the system took to process the transformations on the system.
Times	Contract Creation	Indicates the amount of time that the handler took to determine the contracts to create.
Times	Broker Total	Indicates the total amount of time to process the transaction. This value is the sum of all the fields on the Time tab, and includes: • OnRoute PeopleCode • Inbound Transform • Contract Creation

Copyright © 1988-2007, Oracle. All rights reserved.

Viewing Subscription Contract Handler Statistics

To view subscription contract handler statistics, click the Subscription Contract Handler link on the Statistics page. The Subscription Contract Handler page appears.

The following table describes the fields and data that appear on the Transactions and Times tabs on the Subscription Contract Handler page. This information also applies to the information that appears when you view the data in graphical format.

Tab	Field	Description/Comments
Transactions	All fields	See Common Elements Used in This Section. See Chapter 21, "Using the Service Operations Monitor," Common Elements Used in this Section, page 458.
Times	Notification PeopleCode	Indicates the amount of time the systems took to process notification PeopleCode.
Times	Subscription Contract Process	Indicates internal processing of the contract data.
Times	Subscription Contract Total	Indicates the total amount of time for processing the subscription contract. This value is the sum of processing times for notification PeopleCode and subscription contract processing.

Viewing Publication Contract Handler Statistics

To view publication contract handler statistics, click the Publication Contract Handler link on the Statistics page. The Publication Contract Handler page appears.

The following table describes the fields and data that appear on the Transactions, Publication Contract Total, Publication Handler, Gateway, and Remote PeopleSoft System tabs on the Publication Contract Handler page.

Tab	Field	Description and Comments
Transactions	All fields	See Common Elements Used in This Section. See Chapter 21, "Using the Service Operations Monitor," Common Elements Used in this Section, page 458.

Copyright © 1988-2007, Oracle. All rights reserved.

Tab	Field	Description and Comments
Publication Contract Total	Publication Contract Total	Indicates the total time to process the publication contract. This value is the sum of all fields (except total fields) on all tabs of the Publication Contract Handler page, and includes the following fields: • Publication Contract Handler (Publication Handler tab) • Connectors. (Publication Handler tab) • Outbound Transform. (Publication Handler tab) • Gateway Request Transform (Gateway tab) • Gateway Request Processing (Gateway tab) • Target Connector (Gateway tab) • Gateway Response Miscellaneous (Gateway tab) • Remote Application Server (Remote PeopleSoft System tab) • Remote Database (Remote PeopleSoft System tab)
Publication Handler	Publication Handler	Overall processing time for the publication contract.
Publication Handler	Connectors	Indicates the time to perform the Post to the integration gateway.
Publication Handler	OnSend PeopleCode	Indicates the amount of time that the OnSend PeopleCode event took to run.
Publication Handler	Outbound Transform	Indicates the amount of time to process the outbound transformation.
Publication Handler	OnReceive PeopleCode	Indicates the amount of time that the OnReceive PeopleCode event took to run.
Gateway	Gateway Request Transform	Indicates the amount of time to process the request transform.

Copyright © 1988-2007, Oracle. All rights reserved.

Tab	Field	Description and Comments
Gateway	Gateway Request Processing	Indicates the amount of time that the system performed miscellaneous activities on the integration gateway, including: • Hand-off of the request to the connector. • Authentication. • Writing data to the database.
Gateway	Target Connector	Indicates processing on the target connector.
Gateway	Gateway Response Miscellaneous	Indicates receipt of data from target connector and sending back to the application server.
Gateway	Total	Indicates the total processing time on the integration gateway. This value is the sum of all fields on this tab, and includes: • Gateway Request Transform. • Gateway Request Processing. • Target Connector. • Gateway Response Miscellaneous.
Remote PeopleSoft System	OnRequest PeopleCode	Indicates the amount of time that the OnRequest PeopleCode event took to run on the target system.
Remote PeopleSoft System	Remote Application Server	Indicates processing on the remote application server, and includes: • Authentication. • Serialization and deserialization.
Remote PeopleSoft System	IB Processing Time	Indicates the time spent processing the request. This value does not include database insert processing time, PeopleCode processing time, and so forth.

The following table describes the average total values displayed in the Average Total Time section when you view publication contract handler statistics in graphical format. The fields in the Average Time section are described in the preceding table.

Copyright © 1988-2007, Oracle. All rights reserved.

Field	Description and Comments
Publication Contract Handler	This value is the sum of the following fields located in the Average Time section of the page: • Publication Contract Handler. • Connectors. • Outbound Transform.
Integration Gateways	This value is the sum of the following fields located in the Average Time section of the page: • Gateway Request Transform. • Gateway Request Processing. • Target Connector. • Gateway Response Total.
Remote Integration Service Total	This value is the sum of the following fields located in the Average Time section of the page: • Remote Application Server. • IB Processing Time. •
Publication Contract Total	This value is the sum of all values in the Average Total Time section of the page, and includes: • Publication Contract Handler. • Integration Gateway. • Remote Integration Service Total.

Viewing Inbound Synchronous Service Operation Statistics

To view inbound synchronous service operation statistics, click the Inbound Synchronous link on the Statistics page.

The following table describes the fields and data that appear on the Transactions, Gateway Request Times, PeopleCode Times and Application Server Times tabs on the Inbound Synchronous page. This information also applies to the information that appears when you view the data in graphical format.

Tab	Field	Description and Comments
Transactions	All fields	See Common Elements Used in this Section. See Chapter 21, "Using the Service Operations Monitor," Common Elements Used in this Section, page 458.

 Copyright © 1988-2007, Oracle. All rights reserved.

Tab	Field	Description and Comments
Gateway Request Times	Gateway Request Transform	Indicates the amount of time that the system processed the gateway request transform.
Gateway Request Times	Gateway Request Processing	Indicates the amount of time that the system performed miscellaneous activities on the integration gateway, including: • Hand-off of the request to the connector. • Authentication. • Writing data to the database.
Gateway Request Times	Gateway Request Total	Indicates the total amount of time that the system performed processing on the integration gateway. This value is the sum of the fields on this tab and includes: • Gateway Request Transform. • Gateway Request Processing.
PeopleCode Times	Request Transform	Indicates the amount of time to process the request transform.
PeopleCode Times	OnRoute PeopleCode	Indicates the amount of time to process and execute OnRoute PeopleCode.
PeopleCode Times	OnRequest PeopleCode	Indicates the amount of time to process and execute OnRequest PeopleCode.
PeopleCode Times	Reply Transform	Indicates the amount of time to process the reply transform.
Application Server Times	Application Server Request Processing	Indicates the amount of time the application server takes to process the request.
Application Server Times	Total Application Server Time	Indicates the total processing time on the application server, and is the sum of the following fields: • OnRoute PeopleCode. • OnRequest PeopleCode. • Reply Transform. • Application Server Request Processing.

Copyright © 1988-2007, Oracle. All rights reserved.

Viewing Outbound Synchronous Message Statistics

To view outbound synchronous service operation statistics, click the Outbound Synchronous link on the Statistics page. The Outbound Synchronous page appears.

The following table describes the fields located on the Transactions, Synchronous Total, Synchronous Processing and Gateway Request Times, Remote PeopleSoft Systems and Gateway Response Times tabs on the Outbound Synchronous page.

Tab	Field	Description and Comments
Transactions	All fields	See Common Elements Used in This Section. See Chapter 21, "Using the Service Operations Monitor," Common Elements Used in this Section, page 458.
Synchronous Total	Synchronous Total	Equals the sum of the following tabs: • Synchronous Processing • Gateway Request Times • Remote PeopleSoft System • Gateway Response Times
Synchronous Processing	Outbound Synchronous	Indicates the amount of time to send the service operation.
Synchronous Processing	Connectors	Indicates the time to perform the Post to the integration gateway.
Synchronous Processing	Request Transform	Indicates the amount of time to process and perform the request transform.
Synchronous Processing	OnRoute PeopleCode	Indicates the amount of time to process OnRoute PeopleCode.
Synchronous Processing	OnSend PeopleCode	Indicates the amount of time to process OnSend PeopleCode.
Synchronous Processing	Reply Transform	Indicates the amount of time to process and perform the reply transform.
Gateway Request Times	Gateway Request Connector	Indicates the amount of connector processing time for the request.
Gateway Request Times	Gateway Request Transform	Indicates the amount of time to process and perform the request transform.

Copyright © 1988-2007, Oracle. All rights reserved.

Tab	Field	Description and Comments
Gateway Request Times	Gateway Request Processing	Indicates the amount of time that the system performed miscellaneous activities on the integration gateway, including: • Hand off the request to the connector. • Authentication. • Writing data to the database.
Gateway Request Times	Gateway Request Total	This value is the sum of all fields on the tab, and includes: • Gateway Request Connector. • Gateway Request Transform. • Gateway Request Processing.
Remote PeopleSoft System	OnRequest PeopleCode	Indicates the time to process OnRequest PeopleCode on a remote PeopleSoft system.
Remote PeopleSoft System	Service Operations	Processing on the remote application server, and includes: • Authentication. • Data deserialization and serialization.
Remote PeopleSoft System	Integration Service Total	Indicates the total processing time on remote PeopleSoft systems. This value is the sum of all fields on the tab, and includes: • OnRequest PeopleCode. • Service Operations.
Gateway Response Times	Gateway Response Total	Indicates the total amount of time the system processed the gateway response, including any transformations.

The following table describes the average total values displayed in the Average Total Time section when you view outbound synchronous statistics in graphical (bar chart or pie chart) format. Note that the fields shown in the Average Time section are described in the previous table.

Copyright © 1988-2007, Oracle. All rights reserved.

Field	Description and Comments
Services	This value is the sum of the following fields located in the Average Time section of the page: • Outbound Synchronous. • Connectors. • Request Transform. • OnRoute PeopleCode. • Reply Transform.
Integration Gateways	This value is the sum of the following fields located in the Average Time section of the page: • Gateway Request Connector (target connector). • Gateway Request Transform. • Gateway Request Processing. • Gateway Response Processing.
Integration Service Total	This value is the sum of the following fields located in the Average Time section of the page: • OnRequest PeopleCode. • Service Operations.
Synchronous Total	This value is the sum of all values in the Average Total Time section of the page, and includes: • Services. • Integration Gateways. • Integration Service Total.

Viewing Local Synchronous Service Operation Statistics

To view local synchronous service operation statistics, click the Local Synchronous link on the Statistics page. To view outbound synchronous service operation statistics, click the Outbound Synchronous link on the Statistics page. The Local Synchronous page displays.

The following table describes the fields located on the Transactions, Synchronous Total, Synchronous Processing, Gateway Request Times and Gateway Response tabs on the Local Synchronous page.

Tab	Field	Description and Comments
Transactions	All fields	See Common Elements Used in This Section. See Chapter 21, "Using the Service Operations Monitor," Common Elements Used in this Section, page 458.

 Copyright © 1988-2007, Oracle. All rights reserved.

Tab	Field	Description and Comments
Synchronous Total	Synchronous Total	This field equals the sum of the following tabs: • Synchronous Processing. • Gateway Request Times. • Gateway Response Times.
Synchronous Processing	Local Synchronous	Indicates the amount of time to send the service operation.
Synchronous Processing	Connectors	Indicates the time to perform the Post to the integration gateway.
Synchronous Processing	OnRequest PeopleCode	Indicates the amount of time for processing the OnRequest PeopleCode.
Synchronous Processing	Request Transform	Indicates the processing time for the inbound transform.
Synchronous Processing	OnRoute PeopleCode	Indicates the amount of time for processing the OnRoute PeopleCode.
Synchronous Processing	OnSend PeopleCode	Indicates the amount of time for processing the OnSend PeopleCode.
Synchronous Processing	Reply Transform	Indicates the processing time for the outbound transform.
Gateway Request Times	Gateway Request Connector	Indicates the amount of connector processing time for the request.
Gateway Request Times	Gateway Request Transform	Indicates the amount of time that the system processed the gateway request transform.
Gateway Request Times	Gateway Request Processing	Indicates the amount of time that the system performed miscellaneous activities on the integration gateway, including: • Hand-off of the request to the connector. • Authentication. • Writing data to the database.

Copyright © 1988-2007, Oracle. All rights reserved.

Tab	Field	Description and Comments
Gateway Request Times	Gateway Request Total	Indicates the total processing time for the request on the integration gateway. This value is the sum of the fields on the Gateway Request Time tab and includes: • Gateway Request Connector. • Gateway Request Transform. • Gateway Request Processing.
Gateway Response Times	Gateway Response Total	Indicates the total amount of time the system processed the gateway response, including any transformations.

The following table describes the average total values displayed in the Average Total Time section when you view outbound synchronous statistics in graphical format. Note that the fields shown in the Average Time section are described in the preceding table.

Field	Description/Comments
Services	This value is the sum of the following fields located in the Average Time section of the page: • Local Synchronous. • Connectors. • OnRequest PeopleCode. • Request Transform. • OnRoute PeopleCode. • Reply Transform.
Integration Gateways	This value is the sum of the following fields located in the Average Time section of the page: • Gateway Request Connector (target connector). • Gateway Request Transform. • Gateway Request Processing. • Gateway Response Processing.
Synchronous Total	This value is the sum of all values in the Average Total Time section of the page, and includes: • Services. • Integration Gateways.

Copyright © 1988-2007, Oracle. All rights reserved.

Managing Pub/Sub Server Domains

This section discusses how to work with the Domain Status page and 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 BEA 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.

Page Used to Manage Domain Status

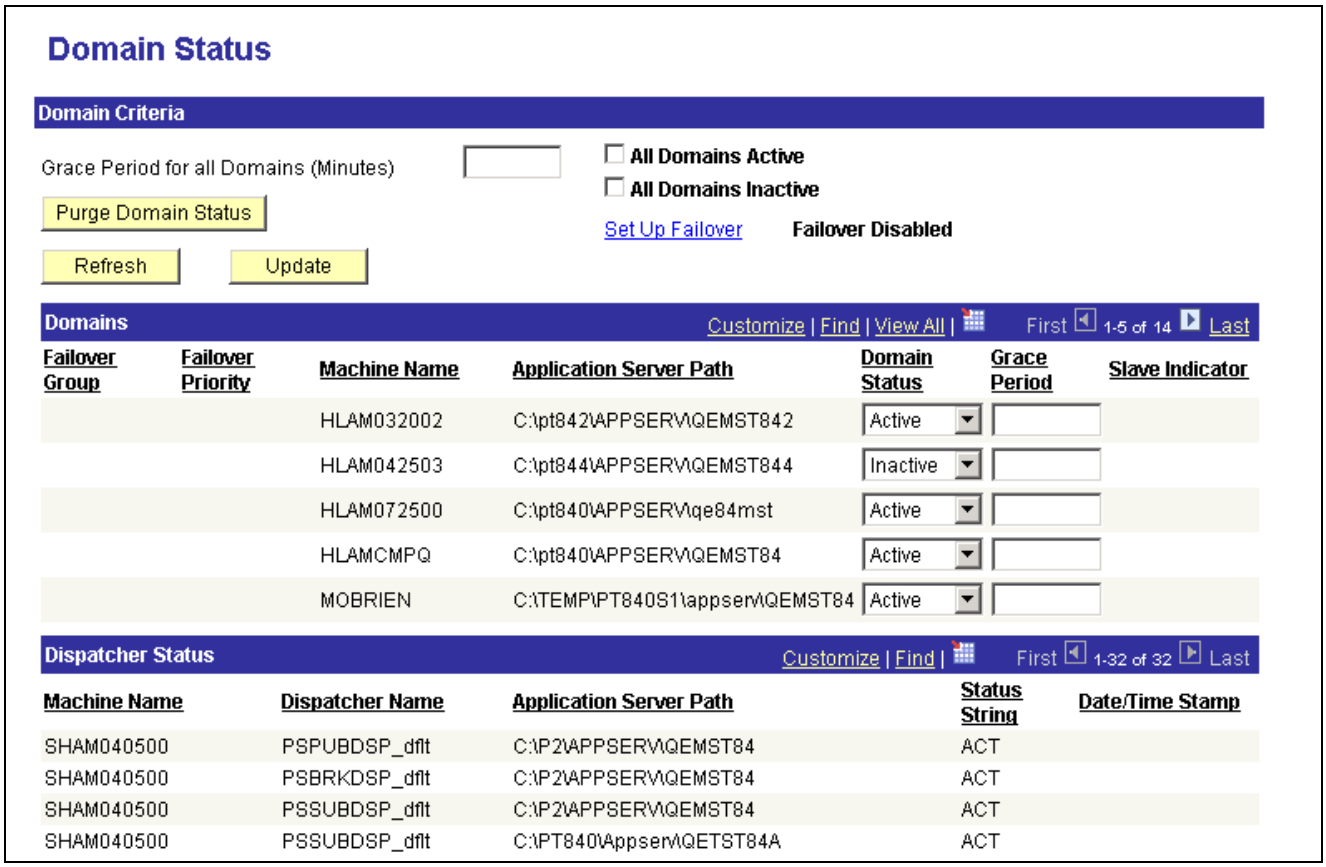
Page Name	Object Name	Navigation	Usage
Domain Status page	AMM_MULTIDOM	Select PeopleTools, Integration Broker, Service Operations Monitor, Monitoring, Domain Status.	Use the Domain Status page to: • View domain status. • Purge domain status. • Change domain status. • Set domain grace periods.

Working with the Domain Status Page

The Domain Status page features three sections, the Domain Criteria section, the Domain Status section, and the Dispatcher Status section.

The following example shows the Domain Status page:

Copyright © 1988-2007, Oracle. All rights reserved.



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 dropdown list 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.

Copyright © 1988-2007, Oracle. All rights reserved.

Viewing Dispatcher Status

The Dispatcher Status section 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 dropdown 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 dropdown 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.

Copyright © 1988-2007, Oracle. All rights reserved.

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.

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.

 Copyright © 1988-2007, Oracle. All rights reserved.

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 groups.

Note. Queue sets within groups must be identical; queue sets between groups must be unique.

The example of the Failover Configuration page in the Setting Up 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, *HLAM032002*, is the first back-up domain as indicated by the failover priority value *1*; the second domain in the list, *HLAM042503*, 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 *HLAM072500*.

The failover priority within group two is as follows: the fourth domain in the list, *HLAMCMPQ*, is the first back-up domain for group two as indicated by the failover priority value *1*; the last domain in the list, *MOBRIEN*, is the second back-up domain as indicated by the failover priority value *2*.

Failover Priorities

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.

Understanding 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.

See Chapter 28, “Tuning Messaging System Performance,” Implementing Master-Slave Dispatchers, page 656.

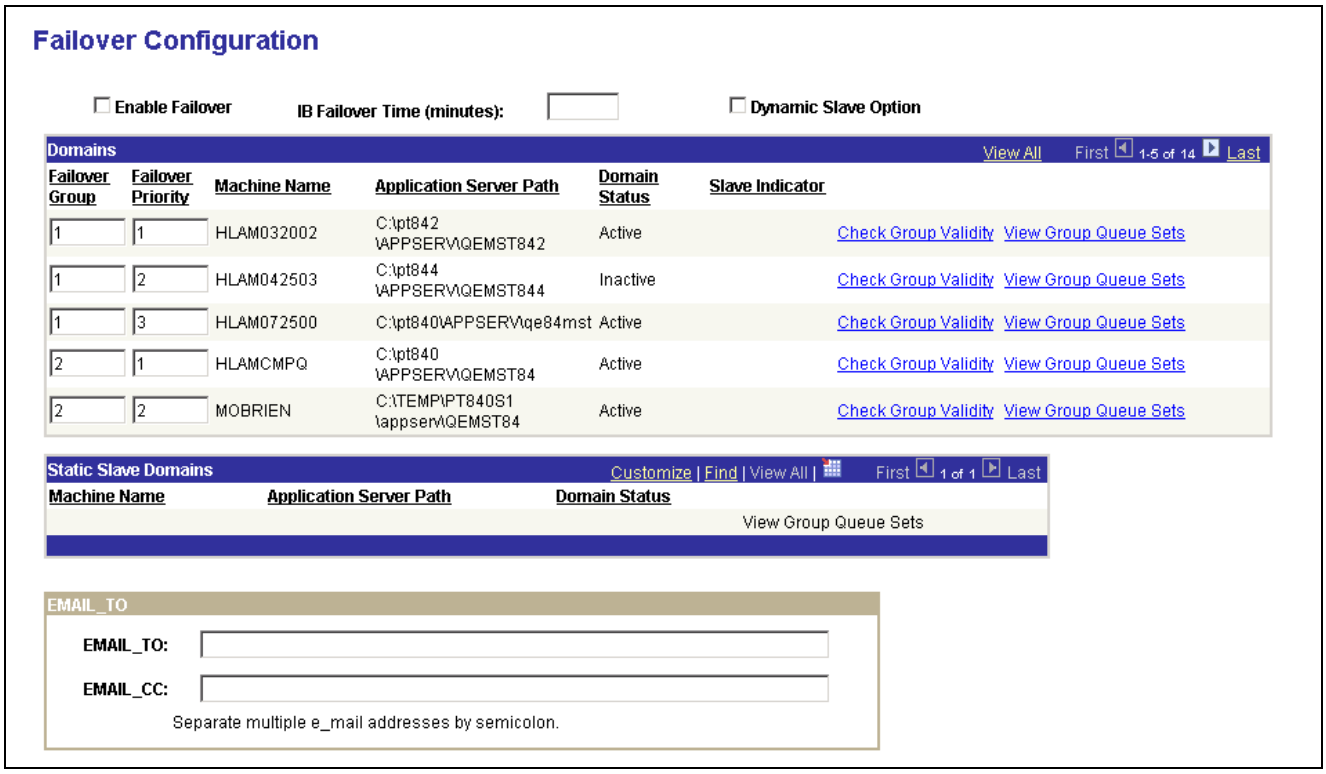
Copyright © 1988-2007, Oracle. All rights reserved.

Page Used to Set Up Domain Failover

Page Name	Object Name	Navigation	Usage
Failover Configuration page	IB_AMM_FAILOVER	Select PeopleTools, Integration Broker, Service Operations Monitor, Administration, Domain Status. The Domain Status page appears. Click the Set Up Failover link.	Use the page to: • Enable failover on domains. • Set up dynamic master-slave dispatchers. • Check queue set validity. • View the queues assigned to failover groups.

Enable Failover on Domains

Use the Failover Configuration page 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.
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.

Copyright © 1988-2007, Oracle. All rights reserved.

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.

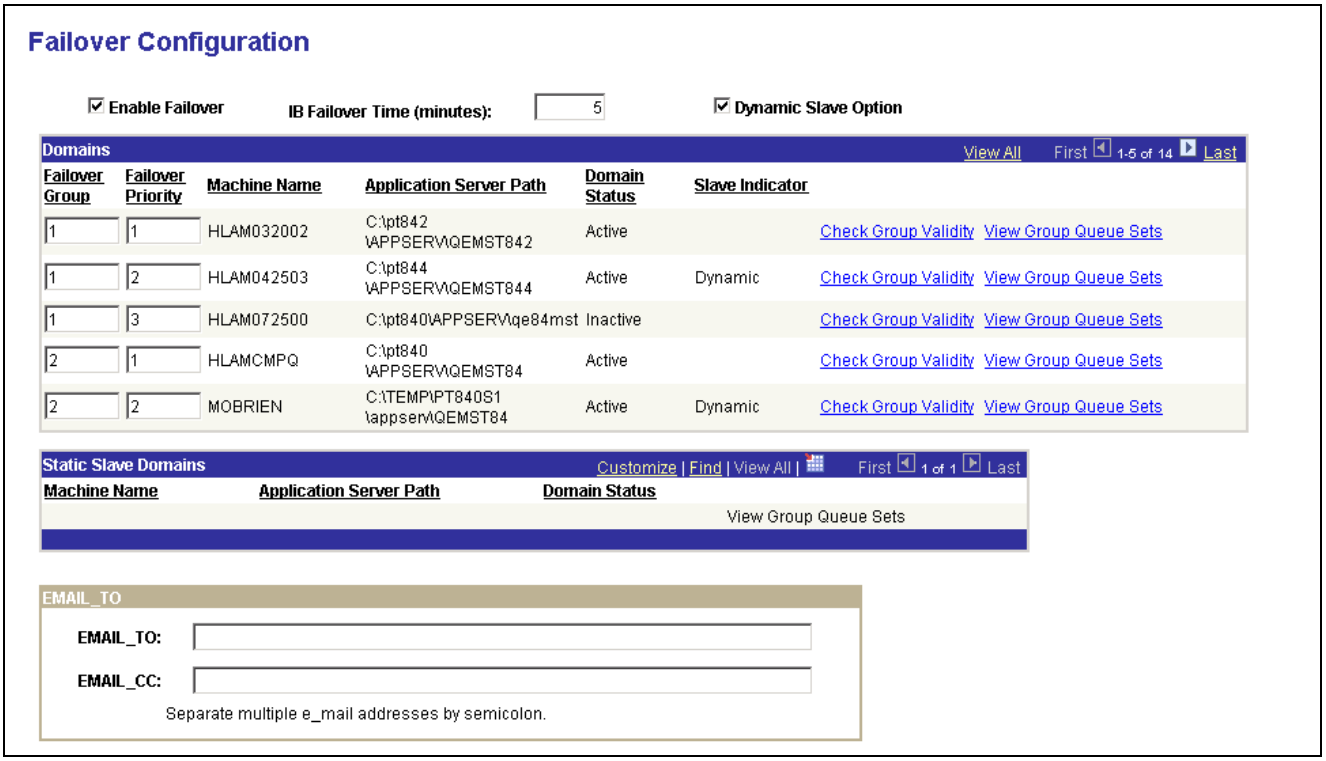
   Separate multiple email addresses with a semicolon.

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:

Copyright © 1988-2007, Oracle. All rights reserved.

- 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 Set 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.

Managing Down Nodes

This section discusses how to:

- View nodes that are down.
- Clear transaction data for system node restart.

Understanding Managing Down Nodes

The Service Operations monitor enables you to view nodes that are down in the integration system and clear transaction data so the system can attempt to restart the node.

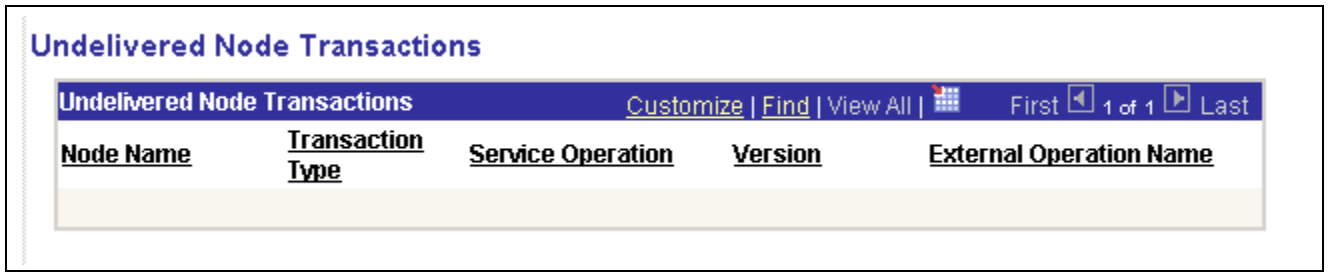
 Copyright © 1988-2007, Oracle. All rights reserved.

Page Used to Manage Down Nodes

Page Name	Object Name	Navigation	Usage
Undelivered Node Transactions page	AMM_NODESDOWN	Select one of the following methods to access the page: • Select PeopleTools, Integration Broker, Service Operations Monitor, Monitoring, Asynchronous Services, Publication Contracts. Click the Transaction Retry Queue link. • Select PeopleTools, Integration Broker, Service Operations Monitor, Administration, Node Status.. Click the Transaction Retry Queue link.	• View information about transactions associated with nodes that are down, including node name, transaction type, service operation and service operation version. • Clear transaction data in the system from down nodes so that the system attempts to restart nodes that are down.

Viewing Transaction Information for Down Nodes

Use the Undelivered Node Transaction page to view information about nodes that are down. The following example shows this page:



Undelivered Node Transaction page

Node Name Name of the node that is down or not responding.

Transaction Type Indicates the transaction type.

Service Operation Indicates the name of the service operation that was being processed by the node when the node stopped responding.

Version Indicates the version of the service operation being processed.

External Operation Name Indicates the name of the service operation sent by the sending node.

Clearing Transaction Data for System Node Restart

Undelivered node transaction information is stored in the Nodes Down table. The Force Retry All button on the Undelivered Node Transaction page enables you to clear the table so that the system can attempt to restart any nodes that are down.

Copyright © 1988-2007, Oracle. All rights reserved.

For example, if a node is in the Nodes Down table and you change the URL of the node, the node becomes free because it is still treated as inoperative (or *down*) based on the old URL. When you click the Force Retry All button, the system retries starting the node.

Click the Force Retry All button on the Undelivered Node Transaction page to clear the Nodes Down table so that the system can restart any nodes that are down.

Pausing, Testing, and Pinging Nodes

This section discusses how to:

- Add a pause time to a local node.
- Delete pause times.
- Test local nodes.
- Ping remote nodes.

Understanding Pausing Nodes

A pause time is an interval of time during which the node becomes inactive. When the pause time begins, the node is shut down until the pause time is scheduled to end.

You might schedule a pause time to perform maintenance tasks or devote server resources to an important batch run. For example, say that you have a complex batch program that runs on the same server as a particular node every Monday morning from 12:05 a.m. to 3:30 a.m. To make sure that the batch program has enough memory devoted to it, you can set a pause time for the node that runs from 12 a.m. to 4 a.m.

During a pause time, transactions are not published or received by the local system. When the system is paused, the node cannot accept service operations sent to it. Consequently, the publishing node must attempt to send transactions again later. The publishing node continues to send transactions until it exceeds the local timeout period. When this happens, the transaction assumes a *Timeout* status in the publisher's queue. The timeout period is an attribute of the publication queue, not the subscription queue.

If the system attempts to send a transaction while the node is paused, the system writes the transaction to the publication and subscription queues, but it cannot publish the transaction until the system is no longer in the paused state.

Note. Pause times do not appear in PeopleSoft Application Designer upgrade projects; you cannot upgrade them.

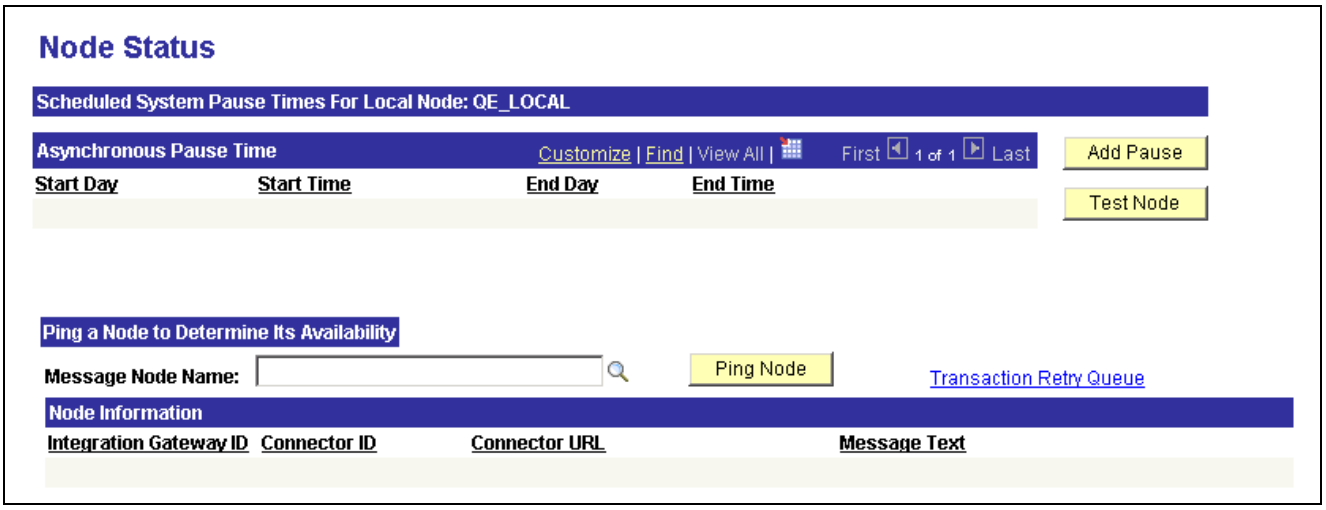
Page Used to Pause, Test and Ping Nodes

Page Name	Object Name	Navigation	Usage
Node Status page	AMM_NODE_STATUS	Select PeopleTools, Integration Broker, Service Operations Monitor, Administration, Node Status..	• Add pause times to local nodes. • Delete pause times. • Test local nodes. • ping remote nodes.

Copyright © 1988-2007, Oracle. All rights reserved.

Adding Pause Times to Local Nodes

Use the Node Status page to add pause time to local nodes. The following example shows the page:



Node Status page

To add a node pause time:

1. Click Add Pause.
2. Select a day of the week in the Start Day dropdown list box.
3. Enter a value in the Start Time edit box.
4. Select a day of the week in the End Day dropdown list box.
5. Enter a value in the End Time edit box.
6. After you have entered the appropriate start and end values to define your pause interval, click OK.

Deleting Pause Times

To delete an existing pause time:

1. In the pause time list, locate the pause time (interval) to delete.
2. Click the Delete button to the right of the entry in the pause time list.

Testing Local Nodes

To test the local node:

1. Make sure you are logged on to the node that you want to test.
2. Click the Test Node button.

Pinging Remote Nodes

A successful ping indicates that the remote node is available to receive transactions. An unsuccessful ping could indicate that the node, gateway, or both are not running.

To ping a remote node:

Copyright © 1988-2007, Oracle. All rights reserved.

1. In the Ping a Node to Determine Availability section, select the node in the Message Node Name dropdown list box to display a list of active nodes.

   The Location column in the grid below reveals the locations defined for the node.

2. Click the Ping Node button.

   The Node Information Section displays connector information defined for the node.

You can also ping remote nodes from the Send Master utility as well as the Simple Post utility.

See *Enterprise PeopleTools 8.49 PeopleBook: PeopleSoft Integration Testing Utilities and Tools*, "Using the Send Master Utility" and *Enterprise PeopleTools 8.49 PeopleBook: PeopleSoft Integration Testing Utilities and Tools*, "Using the Simple Post Utility".

Pausing and Starting Queues

This section discusses how to:

- Pause queues.
- Start queues.

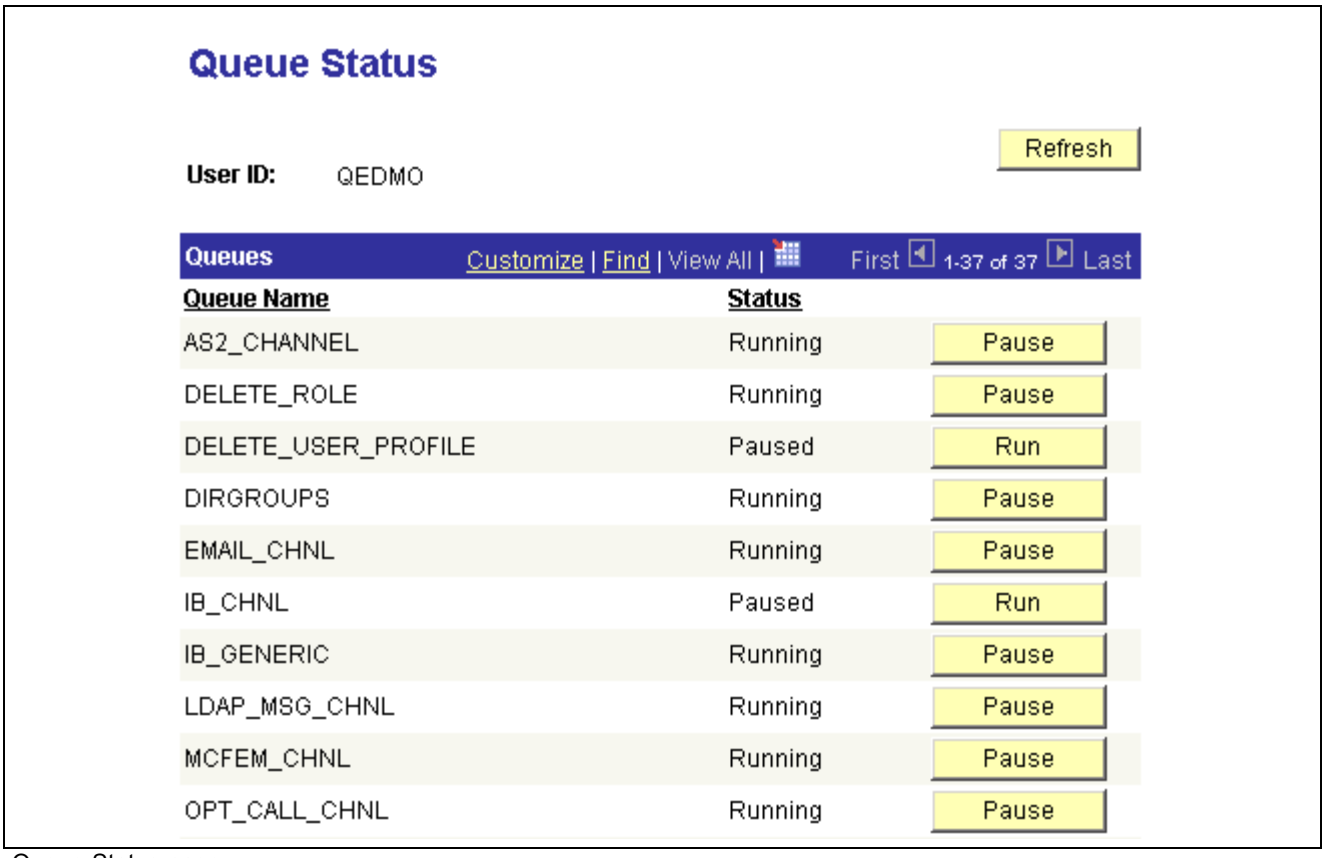
Page Used to Pause and Start Queues

Page Name	Object Name	Navigation	Usage
Queue Status page	IB_MONITOR_QUEUES	Select PeopleTools, Integration Broker, Service Operations Monitor, Administration, Queue Status.	Use this page to pause and start service operation queues.

Pausing Queues

Use the Queue Status page to pause queues on the local database. The following examples shows the page:

 Copyright © 1988-2007, Oracle. All rights reserved.



Queue Status page

Each row in the Queues list displays the queue name and its current status. The label on the button indicates the status to which the queue will change when clicked.

To pause a queue:

1. Select PeopleTools, Integration Broker, Service Operations Monitor, Administration, Queue Status. The Queue Status page appears.
2. In the Queues list, locate the row that contains the queue to pause.
3. Click the Pause button at the end of the row.

Starting Queues

To start a queue:

1. Select PeopleTools, Integration Broker, Service Operations Monitor, Administration, Queue Status. The Queue Status page appears.
2. In the Queues list, locate the row that contains the queue to start.
3. Click the Run button at the end of the row.

Copyright © 1988-2007, Oracle. All rights reserved.

Cleaning Up Orphaned Data From Segment Batch Processing Errors

The section discusses how to clean up orphaned data from segment batch processing errors.

Understanding Cleaning Up Orphaned Data from Segment Batch Process Errors

PeopleTools provides the ability to delete orphaned data left behind after a failed run of a batch segment processing program.

Warning! Perform this clean up only when you are certain that data is orphaned and no segment processing application engine processes are running.

If the batch program is in the middle of processing or if the batch program has abnormally terminated but is to be restarted at a later time, the orphaned data is really not orphaned. Deleting orphaned data in these situations may cause processing problems for the batch program.

See Also

Chapter 12, "Sending and Receiving Messages," Using Restartable Processing for Publishing Large Messages in Batch, page 262

Page Used to Clean Up Orphaned Data from Segment Batch Processing

Page Name	Object Name	Navigation	Usage
Segment Data Cleanup	IB_SEGMENTCLEANUP	Select PeopleTools, Integration Broker, Service Operations Monitor, Administration, Segment Data Cleanup.	Delete orphaned data from segment batch processing errors.

Cleaning Up Orphaned Data from Segment Batch Processing Jobs

Use the Segment Data Cleanup page to clean up orphaned data from segment batch processing jobs. The following example shows this page:

 Copyright © 1988-2007, Oracle. All rights reserved.

Segment Data Cleanup page

Warning! Deleting orphaned data rows can corrupt pending data being processed. Be sure there are no running batch programs that process segment data. Any such program may be adversely affected by deleting orphaned data prematurely.

To clean up orphaned data:

1. Select PeopleTools, Integration Broker, Service Operations Monitor, Administration.
2. Click the Delete Orphaned Data button.

After the system has deleted any orphaned data, it displays a message indicating the deletion is complete.

Using Custom-Defined Components to View Service Operations Data

This section discusses how to:

- Specify the service operation to associate to a custom-defined component.
- Specify custom-defined component parameters.

Understanding Using Custom-Defined Components to View Service Operation Data

You can create a custom component and associate it to a service operation and version. This enables you to navigate to the custom component when you click the Asynchronous Details or Synchronous Details link to view the details for the specified service operation.

Copyright © 1988-2007, Oracle. All rights reserved.

Pages Used for Using Custom-Defined Components to View Service Operations Data

Page Name	Object Name	Navigation	Usage
User Details Component page	PSIBUSERCOMP	Select PeopleTools, Integration Broker, Administration, User Details Component.	• Specify the service operation to associate to the custom-defined component. • Associate the service operation to the custom-defined component.

Specifying Service Operations to Associate to Custom-Defined Components

To specify a service operation to associate to a custom-defined component:

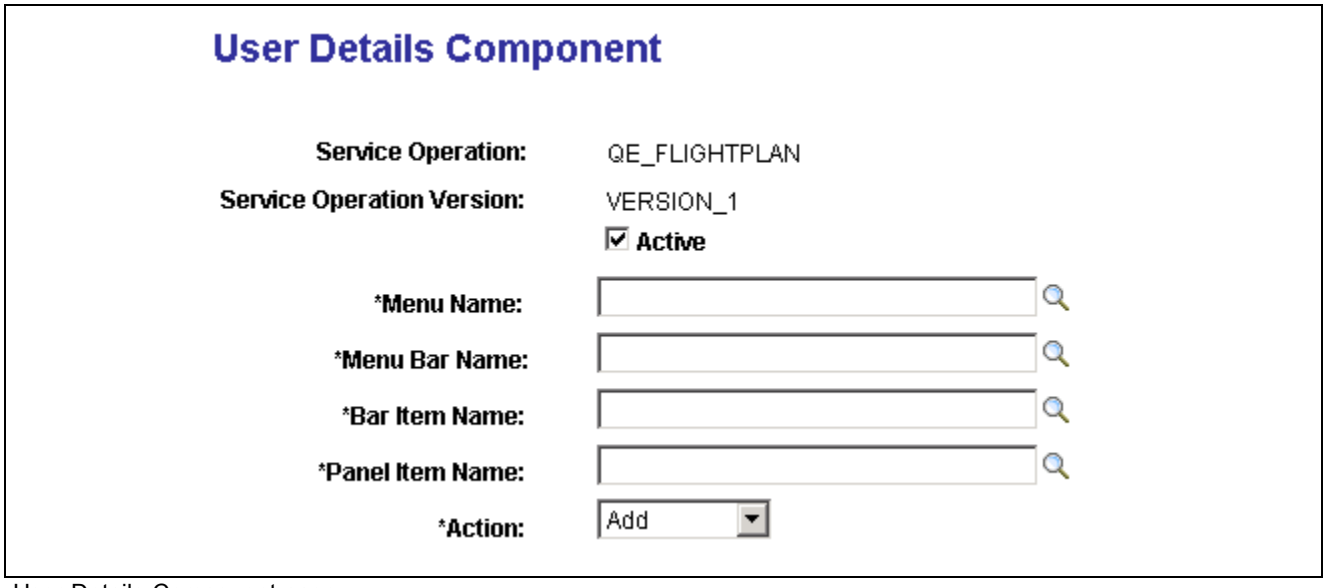
1. Select PeopleTools, Integration Broker, Administration, User Details Component.
2. Click the Add a New Value tab.
3. In the Service Operations field, enter the name of the service operation
4. Click the Add button.

The User Defined Components page appears and you can associate the service operation to the custom-defined component.

Associating Service Operations to Custom-Defined Components

Use the User Details Component to associate a service operation to a custom-defined component.

The following example shows the User Details Component.



User Details Component page

Copyright © 1988-2007, Oracle. All rights reserved.

Active Indicates if the component is active. Clear the box to inactivate the component.

By default the component is active.

Menu Name From the dropdown list box, select the menu name where the page is located.

Menu Bar Name From the dropdown list box, select the menu bar name where the page is located.

Bar Item Name From the dropdown list box, select the bar item name.

Panel Item Name From the dropdown list box, select the page name.

Action From the dropdown list box, select the action for the page. The valid values are:

- *Add.*

  Select to add a new high-level key, such as a new employee ID or customer. Except in the case of effective dating, Add is used to insert a new current row or to update future rows.

- *Corr.* (Correction.)

  Select to update any rows (history, current, and future) in an effective-dated record. Use only with effective-dated records. This is translated to correct history at runtime.

- *Up/Dsp All.* (Update/Display All.)

  Select to update current and future rows in an effective-dated record. Use only with effective-dated records. Do not use these actions unless the main record that is associated with the page definitions is effective-dated. This is translated to include history at runtime.

- *Upd/Display.* (Update/Display.)

  Select to update existing rows only.

Purging Runtime Monitor Tables

The PeopleSoft system provides a collection of Data Mover scripts that you can run to purge the runtime Service Operations Monitor tables within a database. These scripts reside in the PS_HOME\scripts directory on your file server. The following table describes the purpose of each script.

Warning! Shut down the application server before running any of the Data Mover scripts described in this section.

Copyright © 1988-2007, Oracle. All rights reserved.

Script Name	Description
AppMsgPurgeAll.dms	Deletes queue data from every archive or live runtime Service Operations Monitor table in the database, regardless of status. Typically, you run this script after an upgrade or while switching from a demonstration to a production environment.
AppMsgPurgeArchive.dms	Deletes queue data from every archive runtime Service Operations Monitor table in the database.
AppMsgPurgeLive.dms	Deletes queue data from every live runtime Service Operations Monitor table in the database.

Using the Services Operations Monitor Component Interface

The Service Operations Monitor includes a collection of inquiry methods that you can access with a component interface.

Use the MSGSTATUSSUMMARY component interface to extract information from the Service Operations Monitor. The output of the component interface reveals the amount of contracts that are in the queue. The contracts appear grouped by status and service operation or grouped by status and queue.

You can use the following user-defined methods to extract information:

- FillPubConByMsg()
- FillPubConByChannel()
- FillSubConByMsg()
- FillSubConByChannel()

Beginning with PeopleTools 8.48 queues replaced channels from earlier PeopleTools 8.4x versions. As a result, once you have a rowset object pointing to ByChannel, reference QUEUENAME when working with the code.

The following example shows ASP code that accesses the MSGSTATUSSUMMARY component interface with COM.

```
'Create a peoplesoft session
Set oSession = server.CreateObject ("PeopleSoft.Session")
nStatus = oSession.Connect(1, oConnectString, oUserName, oPassword,0)

'Get the skeleton of the APPMSGMON CI
Set oCI = oSession.GetCompIntfc("MSGSTATUSSUMMARY")

'get an instance of the CI
nStatus = oCI.Get()
'execute the method to fill the collection

If oChoice = 1 then
   nStatus = oCI.FillPubConByChannel()
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
   'Set oRows to the properties collection
   Set oRows = oCI.PubConByChannel
End If

If oChoice = 2 then
   nStatus = oCI.FillPubConByMsg()
   'Set oRows to the properties collection
   Set oRows = oCI.PubConBymsg
End If

If oChoice = 3 then
   nStatus = oCI.FillSubConByChannel()
   'Set oRows to the properties collection
   Set oRows = oCI.SubConByChannel
End If

If oChoice = 4 then
   nStatus = oCI.FillSubConByMsg()
   'Set oRows to the properties collection
   Set oRows = oCI.SubConByMsg
End If
```

See Also

Enterprise PeopleTools 8.49 PeopleBook: PeopleSoft Component Interfaces, "Introducing Component Interfaces"

Using PeopleCode to Read and Write Errors to the Asynchronous Error Queue

PeopleSoft provides the following two methods to read and write errors to the asynchronous error queue:

GetMessageError	If an error occurs during processing of a service operation instance, publication contract or subscription contract, the error is read from the appropriate queue .
SetMessageError	If an error occurs during processing of a service operation instance, publication contract or subscription contract, the error is written to the appropriate queue .

See Also

Enterprise PeopleTools 8.49 PeopleBook: PeopleCode API Reference, "PeopleCode API Reference Preface"

Copyright © 1988-2007, Oracle. All rights reserved.

 Copyright © 1988-2007, Oracle. All rights reserved.

CHAPTER 22

Managing Error Handling, Logging, Tracing, and Debugging

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

- Manage integration gateway message and error logging.
- Manage application server logging and tracing.
- Debug integrations.

Understanding Error Handling, Logging, Tracing and Debugging

Error handling, logging, tracing, and debugging with PeopleSoft Integration Broker can occur on an integration gateway, application server, or application engine, depending on the type and location of processing.

Understanding Integration Gateway Error Handling

Error handling is an integration gateway service that assists connectors to manage errors that occur during processing. Errors on the integration gateway are handled by target connectors and listening connectors.

This section discusses:

- Target connector error handling.
- Listening connector error handling.
- Integration gateway exception types.

Target Connector Error Handling

The Target Connector Interface (TCI) specifies the methods that target connectors must implement for the integration gateway to manage them. These methods include a set of standard exceptions that target connectors generate when they experience errors during processing.

Listening connectors or the gateway manager catch these exceptions and provide an appropriate implementation for each. When the source of the message is an integration engine, the gateway manager catches the exceptions. Otherwise, listening connectors are responsible for handling exceptions that are generated during processing.

Copyright © 1988-2007, Oracle. All rights reserved.

Listening Connector Error Handling

Unlike target connectors, listening connectors are not managed by the gateway manager and, therefore, do not adhere to any interface. However, a listening connector must invoke the gateway manager to pass a message from the integration gateway to the integration engine.

The gateway manager has predefined exceptions.

In general, exceptions are thrown in a target connector and caught by a listening connector. As a result, a listening connector must catch these exceptions and handle them as appropriate. Typically, the listening connector generates an error message and sends it back to the requester.

Integration Gateway Exception Types

This section discusses integration gateway exception types.

Standard Exceptions

The following standard error and exception types are handled by the integration gateway, target connectors, and listening connectors:

Exception Type	Description
DuplicateMessageException	A target connector attempted to process a message that has already been processed. This is usually discovered based on an error that is attained from the external system that is being contacted. Of the connectors that are delivered with the PeopleSoft software, only the PeopleSoft 8.1 target connector (PSFT81TARGET) can generate this exception. Target connectors are not required to generate this exception.
ExternalApplicationException	The message reached its intended destination but could not be processed. Determining that the destination could not process a message requires significant knowledge of the destination system, which a target connector might not have. Whenever possible, a target connector should attempt to determine this situation; otherwise this task must be decentralized and handled outside of the integration gateway. For example, the HTTP target connector (HTTPTARGET) generates this exception when the external system returns an HTTP system code of *500*.
ExternalSystemContactException	The target connector cannot establish a connection with the intended destination. This is one of the most common exceptions. When this exception is thrown during an asynchronous transaction, PeopleSoft Integration Broker tries to resend the message until successful.
GeneralFrameworkException	A general error occurred.
InvalidMessageException	A connector or the gateway manager determined that the message cannot be processed because of missing or erroneous information in a request or response.

 Copyright © 1988-2007, Oracle. All rights reserved.

Exception Type	Description
MessageMarshallingException	A gateway service's attempt to get information from an IBRequest or IBResponse failed. This can occur when the gateway services attempt to access a content section of a document by using an out-of-range index from one of the following methods: • GetContentSectionAt(index) • GetContentSectionInfoAt(index) • RemoveContentSectionAt(index) If you try to access IBRequest or IBResponse with an out-of-range index by using any of these methods, this exception is thrown automatically and processing is interrupted.
MessageUnmarshallingException	A gateway service's attempt to build an IBRequest or IBResponse failed. Failure can occur when: • Instantiating an IBRequest or IBResponse from a Multipurpose Internet Mail Extensions (MIME) format where the message that was sent does not comply with the PeopleSoft MIME format. • Instantiating an IBRequest by using the PS_XML format and passing an invalid PS_XML message. This is typically from the HTTP listening connector. • Setting invalid values to methods, such as setTransactionID or setMessageType. These failures cause the integration gateway to generate this exception automatically and processing is interrupted.

Java Exceptions

Target connectors and listening connectors can handle miscellaneous Java exceptions, such as NullPointerException and ArrayOutOfBoundsException.

Managing Integration Gateway Message and Error Logging

This section provides an overview of message and error logging and discusses how to:

- Set up message and error logging.
- View non-English characters in integration gateway log files.
- Manage message logging.
- Manage error logging.

Understanding Message and Error Logging

Error and message logging is a gateway service that you use to monitor messages that flow through the integration gateway.

Copyright © 1988-2007, Oracle. All rights reserved.

Logging takes place within both target and listening connectors. Connectors can log all message requests and responses. As a result, you can use logging to:

- Track message flow.
- Troubleshoot processing errors.

Setting Up Message and Error Logging

By default, an integration gateway logs all errors and warnings, as well as information of important, standard, and low importance.

Set up message and error logging by using the integrationGateway.properties file. Use the Logging Setting section to view or change default settings, such as the level of gateway logging, where the system writes log files, the maximum size of the log file, and the number of file backups or archives to keep.

See Also

Chapter 7, "Managing Integration Gateways," Setting Logging Properties, page 72

Viewing Non-English Characters in Integration Gateway Log Files

To view non-English characters in integration gateway log files, enable UTF-8 encoding in your web browser. For example, if you are using Microsoft Internet Explorer 5.5, you can enable UTF-8 encoding by selecting View, Encoding, Unicode (UTF-8). If you are using Netscape Navigator 6.0, you can enable UTF-8 encoding by selecting View, Character Encoding, Unicode (UTF-8).

Managing Message Logging

Message logging records the following information for messages that pass through the integration gateway:

- Time and date.
- Message description.
- Content of the passed message object.
- Message level.

The default location of the integration gateway message log is <PS_HOME>\webserv\<DOMAIN>\applications\peoplesoft\PSIGW\msgLog.html.

Change the location of the log in the integrationGateway.properties file.

Message Logging in Target Connectors

Message logging in a target connector occurs:

- Before delivering the request to the external system.

  The connector logs the request in the format in which the external system delivered it. For example, an HTTP target connector logs the exact HTTP output stream request. The PeopleSoft target connector logs the MIME request to be sent to the integration engine.

- After it receives a response from the external system.

  The connector logs the response in the format in which it receives it. For example, an HTTP target connector logs the exact HTTP input stream response. The PeopleSoft target connector logs the MIME response that it received from the integration engine.

 Copyright © 1988-2007, Oracle. All rights reserved.

Message Logging in Listening Connectors

Message logging in a listening connector occurs:

- At the point where the request enters the system.

  The connector logs the request in the format in which the sending system delivers it. For example, the HTTP listening connector logs the exact HTTP input stream request. The PeopleSoft listening connector logs the MIME request that it received from the integration engine.

- Following the delivery of a response to the requestor system.

  The connector logs the response in the format in which it was delivered. For example, the HTTP listening connector logs the exact HTTP output stream response. The PeopleSoft listening connector logs the MIME response that it sent back to the integration engine.

Message Logging Methods and Parameters

Invoke the logMessage method for integration gateway message logging:

```
logMessage(String Description, Object message, int MessageLevel)
```

Use the following parameters:

Parameter	Description
Description	Specify a description as a string.
Object	Specify the message object. Typically this object is an IBRequest or IBResponse. If another object is passed, the toString method is invoked for the object, and the result is logged.
MessageLevel	Set the relative importance of the information that you are logging. The ig.log.level property setting in the integrationGateway.properties file determines the log level that is currently in effect. If the MessageLevel value that is passed here is less than or equal to the ig.log.level property setting, the message is written to the log file. Values are: • *3*: Important information. • *4*: Standard information. • *5*: Low-importance information. The ig.messageLog.filename property in the integrationGateway.properties file determines the log file location.

Managing Error Logging

Error logging captures processing errors that occur in the integration gateway. When an error occurs, the following information is logged:

- Error level.
- Description.

Copyright © 1988-2007, Oracle. All rights reserved.

- Message catalog entry information.
- Stack trace identifying the problem.
- IBRequest and IBResponse (if available).

The default location of the integration gateway error log is <PS_HOME>\webserv\<DOMAIN>\applications \peoplesoft\PSIGW\errorLog.html

Change the location of the log in the integrationGateway.properties file.

Error Logging Methods and Parameters

Invoke the logError method for integration gateway error logging:

```
logError (String Description, IBRequest, IBResponse, int ErrorLevel, Throwable)
```

Use the following parameters:

Parameter	Description
Description	Specify a description as a string.
IBRequest	Specify the IBRequest for this transaction, if available. If not available, pass Null.
IBResponse	Specify the IBResponse for this transaction, if available. If not available, pass Null.
ErrorLevel	Specify whether the log is written to permanent storage. This determines the severity of the error. The ig.log.level property in the integrationGateway.properties file determines the log level that is currently in effect. If the ErrorLevel value that is passed here is less than or equal to the ig.log.level property setting, the error is written to the log file. Values are: • *100*: Language exception. • *1*: Standard gateway exception. • *2*: Warning. The ig.errorLog.filename property in the integrationGateway.properties file determines the log file location.
Throwable	Specify the Java exception or error that is associated with the error. This is used to log the stack trace that is associated with the error.

The gateway manager and delivered listening connectors feature built-in error logging that invokes the logError method. The delivered target connectors do not feature built-in error logging, and instead generate errors to the gateway manager or listening connectors, where they are handled or logged.

 Copyright © 1988-2007, Oracle. All rights reserved.

Managing Application Server Logging and Tracing

Use the PeopleSoft Application Server Administration menu to:

- View application server and BEA Tuxedo log files.

  See *Enterprise PeopleTools 8.49 PeopleBook: System and Server Administration*, "Using PSADMIN Menus," Editing Configuration and Log Files.

- Trace Structured Query Language (SQL) and PeopleCode on your domains.

  See *Enterprise PeopleTools 8.49 PeopleBook: System and Server Administration*, "Setting Application Server Domain Parameters," Trace Options.

- Set the level of network tracing (log fence).

  See *Enterprise PeopleTools 8.49 PeopleBook: System and Server Administration*, "Setting Application Server Domain Parameters," Domain Settings.

- View the certificate authentication logs, including information about mismatched distinguished names and certificates that are not in the database.

  This information is contained in the APPSRV.LOG file.

You can also use the tracing functionality in PeopleSoft Application Engine, which enables you to monitor the performance of transforms in your implementation of PeopleSoft Integration Broker.

See Also

Chapter 20, "Applying Filtering, Transformation and Translation," page 373

Enterprise PeopleTools 8.49 PeopleBook: PeopleSoft Application Engine, "Tracing Application Engine Programs"

Debugging Integrations

This section discusses how to:

- Debug handler PeopleCode.
- Handle common issues.

Debugging Handler PeopleCode

Use the Handler Tester utility to debug service operation handler PeopleCode.

The Handler Tester utility enables you to use the PeopleSoft Pure Internet Architecture to test any of the following handler types:

- OnSend.
- OnRequest
- OnRouteReceive
- OnRouteSend.

Copyright © 1988-2007, Oracle. All rights reserved.

- OnAckReceive
- OnNotify.

You can test handlers without setting up a routing definition, without having pub/sub booted on your application server, and without impacting other developer activity on the system.

See Also

Enterprise PeopleTools 8.49 PeopleBook: PeopleSoft Integration Testing Utilities and Tools, "Using the Handler Tester Utility"

Handling Common Issues

Use this table to handle common issues in PeopleSoft Integration Broker:

Area or Suspected Issue	Debugging Suggestion
Application server exceptions.	Check the application server log: <PS_HOME>\appserv\<Domain>\LOGS\ appsrv.log
Message handlers are not running.	Check the application server domain status or queue status in the PeopleSoft Application Server Administration menu (PSADMIN). Select Domain Status, Server Status or Domain Status, Queue Status.
Integration gateway.	Check the integrationGateway.properties file and verify the property settings. The default file location is <PS_HOME>\webserv \<DOMAIN>\applications\peoplesoft\PSIGW\WEB-INF \integrationGateway.properties.
Integration gateway.	Check the integration gateway message log. The default file location is <PS_HOME>\webserv\<DOMAIN> \applications\peoplesoft\PSIGW\msgLog.html.
Queues are paused.	Check the Service Operations Monitor. Select PeopleTools, Integration Broker, Service Operations Monitor, Administration, Queue Status.
A node is paused.	Check the Service Operations Monitor. Select PeopleTools, Integration Broker, Service Operations Monitor, Administration, Node Status.
Incorrect gateway uniform resource locator (URL).	Check the Gateways component to verify that the integration gateway URL is correct. Select PeopleTools, Integration Broker, Configuration, Gateways.
Node inactive.	Check the node definition. Select PeopleTools, Integration Broker, Integration Setup, Nodes.

Copyright © 1988-2007, Oracle. All rights reserved.

Area or Suspected Issue	Debugging Suggestion
Subscription PeopleCode is missing or incorrect.	Check the Service Operations Monitor. Select PeopleTools, Integration Broker, Monitoring, Asynchronous Services, Subscription Contracts..
A service operation is inactive.	Check the service operation definition in the PeopleSoft Pure Internet Architecture. Select PeopleTools, Integration Broker, Integration Setup, Service Operations.
There are transform problems.	• Check the Application Engine object in PeopleSoft Application Designer. • For before and after images, check the Service Operations Monitor. For asynchronous service operations, , select PeopleTools, Integration Broker, Service Operations Monitor, Monitoring, Asynchronous Details. Click the View XML link for the publication contract or subscription contract. For synchronous service operations, select PeopleTools, Integration Broker, Service Operations Monitor, Monitoring, Synchronous Details. Use the Log Type dropdown list box to select Request Transformed or Response Transformed, and then click View XML. • Verify that the TraceAE flag in the following directory equals 8192: <PS_HOME>\appserv\<Domain>\psappsrv.cfg Setting the TraceAE flag in the psappsrv.cfg file instructs the application server to generate a transformation trace log with the .aet extension, written to the following directory: <PS_HOMEy>\appserv\<Domain>\LOGS\ <operID>_<machine name>.AET The log file contains: - The original XML structure as it entered the transformation engine. - The output of the XML as it passed through each step of the transform program.

Copyright © 1988-2007, Oracle. All rights reserved.

Copyright © 1988-2007, Oracle. All rights reserved.

CHAPTER 23

Providing Services

This chapter discusses how to:

- Provide services.
- Access generated WSDL documents.
- Delete WSDL documents.

Understanding Providing Services

This section provides an overview of using the PeopleSoft Integration Broker to provide web services and generate WSDL documents.

PeopleSoft Integration Broker features a Provide Web Service wizard that steps you through the task of providing web services.

Understanding the Provide Web Service Wizard

This section provides an overview of the Provide Web Service wizard.

Features of the Provide Web Service Wizard

The Provide Web Service component (IB_WSDL_EXPORT) features a wizard you can use to provide web services. You can publish WSDL document to the WSDL repository in the PeopleSoft system or external UDDI repositories.

After you generate a WSDL document, the Provide Web Service wizard display a WSDL URL for each document you generated. This enables you to access the WSDL document content using the WSDL URL. In addition, PeopleSoft Integration Broker provides a WSIL URL which lists the provided services and corresponding WSDL URLs.

You can use the Provide Web Service wizard to select one or more services for which to generate WSDL documents. The system generates a separate WSDL document for each service.

Other features include:

- Supports WS-interoperability standards for WSDL.
- Provides WSDL version 1.1 documents.
- Provided WSDL documents include WS-Addressing header elements for asynchronous request/response operation types.

Copyright © 1988-2007, Oracle. All rights reserved.

- Provided WSDL documents include WS-Security elements . UsernameToken type is supported.
- Provided WSDL documents include PartnerLinkType elements, which are used when consumed by a BPEL application.

Operation Types Supported

The Provide Web Service wizard can create WSDL documents for service operations having the following operation types:

- Synchronous.
- Asynchronous one-way.
- Asynchronous request/response.

Requirements for Nonrowset-Based Message Schemas

This section discusses requirements and considerations for creating nonrowset-based message schemas for service operations in order to generate WSDL documents using the Provide Web Service wizard.

Note. The PeopleSoft system automatically generates message schemas for rowset-based messages.

Target Namespace

Nonrowset-based message schemas must contain a target namespace. If no target namespace exists in the schema an error occurs when the system generates the WSDL document.

You may define multiple schema imports to the same target namespace, but different schema locations must be defined.

Multiple Root Element and Complex Type Tags

If the PeopleSoft system finds multiple root <element> or <complexType> tags in nonrowset-based message schemas, only the first one is referenced in the WSDL document or container message schema.

In addition, the WSDL would allow schema imports to the same target namespace but different schema locations and use <xsd:include> when the schema Namespace is same as the WSDL namespace.

See Also

Chapter 23, “Providing Services,” Prerequisites for Providing Services, page 521

Locations for Publishing WSDL Documents

Using the Provide Web Service wizard, you can publish WSDL documents to the follow locations:

- PeopleSoft WSDL repository. (Application database.)

  The PeopleSoft WSDL repository is the default publishing location. All generated WSDL documents are published to the PeopleSoft WSDL repository. You may publish WSDL documents to a UDDI repository in addition to the WSDL repository.

- Universal Description, Discovery, and Integration (UDDI) repositories.

  Services published to UDDI repositories are available to other PeopleSoft and external systems. If another PeopleSoft system wants to invoke an exported service from UDDI, it can consume the WSDL document from the UDDI reference into its system to create a service and routing definition.

 Copyright © 1988-2007, Oracle. All rights reserved.

UDDI Repositories and Endpoints

When publishing a WSDL document to a UDDI repository, the PeopleSoft system publishes the current endpoint value defined in the Target Location field in the Service Configuration page.

The endpoint value in the actual WSDL document is a dynamic one, since you can change the target location value, for example, when you move from development to production.

If you change the target location you should change the endpoints of previously published WSDL documents, either manually in the UDDI registry or by republishing the WSDL documents to your UDDI repository using the changed endpoint.

WSDL URL Formats

After you a publish a WSDL document using the Provide Web Service wizard, the system displays a WSDL URL. The URL provided is the path to the WSDL document location in the WSDL repository in the PeopleSoft Pure Internet Architecture. The URL is used by external systems that will be invoking a PeopleSoft service.

The default URL format is path style. The following example shows a WSDL URL in path format:

```
http://localhost/PSIGW/PeopleSoftServiceListeningConnector/PT_WORKLIST.1.wsdl
```

The path style URL is generated by appending the WSDL document name to the target location value specified in the Service Configuration page.

PeopleSoft Integration Broker also supports a query parameter format. The following example shows a WSDL URL in query parameter format:

```
//PeopleSoftServiceListeningConnector?Operation=GetWSDL&wsdl=PT_WORKLIST.1
```

The query parameter style URL is generated by passing either the WSDL document name or service name.version or service alias.version as a query parameter.

PeopleSoft still supports the query parameter format, however path format is preferred. Note that if using query parameter format, manually intervention may be required if the schema target location is changed.

Provided WSDL Documents

Every WSDL document you generate using PeopleSoft Integration Broker is divided into sections. This section describes the WSDL document sections and provides an example of the WSDL template that the PeopleSoft system uses to generate WSDL, as well as example WSDL documents for each of the supported operation types.

Sections of Provided WSDL Documents

WSDL documents that you provide using PeopleSoft Integration Broker contain the following sections:

Copyright © 1988-2007, Oracle. All rights reserved.

Section	Description
<definitions>	Specifies the namespaces for the WSDL document, W3C XML Schema and SOAP. A unique namespace will be captured from the Service definition, which will be used to define the WSDL namespaces. The format of this namespace is as follows: http://xmlns.oracle.com/Enterprise/<App Name>/<Service Name>. When a service is defined within an application database, the namespace field is defaulted to the service namespace defined on the Service Configuration page.
<partnerLinktype>	A partnerLinkType defines the role of services and the port Type.
<types>	Captures the simple and complex types required by the schema of the request and response message definitions of the service operations. For services with component interface handlers, some of the system methods, such as Create and Get, will require complex message types resembling the structure of the component interface buffer.
<message>	Defines the abstract messages required for the selected operations. The data types for these are obtained from the Types section of the WSDL document.
<portType>	Features a named set of abstract operations and the abstract messages involved. This section includes all operations of the service selected for export.
<binding>	Specifies the network protocol and data format of messages used for a specific port type. For providing web services PeopleSoft utilizes SOAP packaging and HTTP transport protocols. The data format of messages is the Document style format.
<operation>	This is an abstract definition of a service operation, which specifies request/response/fault messages.
<service>	A service groups together endpoints that implement a common interface.

Note. In WSDL documents generated by the PeopleSoft system, WS-Security policies are assigned to the bind operation section.

Example 1: WSDL Template

The following example is the WSDL document template that the PeopleSoft system uses when it generates WSDL documents. The elements in bold are WSDL document sections discussed in the previous section:

```
definitions name="DefinitionsName"
  targetNamespace="NamespaceURI"
  xmlns:prefix="NamespaceURI"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns="http://schemas.xmlsoap.org/wsdl/">
  <plnk:partnerLinkType name="PartnerLinkTypeName">
    <!-- Provider Role -->
       <plnk:role name="ProviderRoleName">
       <plnk:portType name="ProviderPortTypeReference">
        </plnk:role>
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
  <!-- In case of Async Request/Response this role is also required -->
  <plnk:role name="RequestorRoleName">
    <plnk:portType name="CallbackPortTypeReference">
     </plnk:role>
    </plnk:partnerLinkType>
<types>
    <!-- One or more schemas -->
  </types>
  <message name="MessageName">
    <part name="PartName" type="TypeNameReference"/>
  </message>
  <portType name="PortName">
    <operation name="OperationName">
      <input message="MessageNameReference"/>
      <output message="MessageNameReference"/>
      <fault message="MessageNameReference"/>
    </operation>
  </portType>
  <binding name="BindingName" type="PortNameReference">
    <soap:binding style="rpc|document"
      transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="OperationName">
      <soap:operation soapAction="ActionValue"/>
      <input>
        <soap:body
          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          namespace="TargetNamespace"
          use="encoded"/>
     </input>
     <output>
      <soap:body
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        namespace="TargetNamespace"
        use="encoded"/>
     </output>
    </operation>
  </binding>
  <service name="ServiceName">
    <port name="PortName" binding="BindingNameReference">
      <soap:address location="URL"/>
   </port>
  </service>
</definitions>
```

Example 2: Synchronous WSDL Example

The following example shows a synchronous WSDL document provided by the PeopleSoft system:

```
<?xml version="1.0" ?>
<wsdl:definitions name="PT_WORKLIST.1"
  targetNamespace="http://xml.namespace.oracle.com/services"
```

Copyright © 1988-2007, Oracle. All rights reserved.

```
xmlns:GetWorklistEntryStatusRequest.v1="http://xmlns.oracle.com/
 Enterprise/Tools/schemas/PT_WL_GET_INSTANCE_REQ_CONT.v1"
xmlns:GetWorklistEntryStatusResponse.v1="http://xmlns.oracle.com/
 Enterprise/Tools/schemas/PT_WL_GET_INSTANCE_RESP_CONT.v1"
xmlns:OperationFault.V1="http://xmlns.oracle.com/schemas/Fault"
xmlns:plnk="http://schemas.xmlsoap.org/ws/2003/05/partner-link/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http:
 //xml.namespace.oracle.com/services"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsp="http://schemas.xmlsoap.org/ws/2002/12/policy">
<wsp:UsagePolicy wsdl:Required="true" />
<plnk:partnerLinkType name="PT_WORKLIST_PartnerLinkType">
  <plnk:role name="PT_WORKLIST_Provider">
    <plnk:portType name="tns:PT_WORKLIST_PortType" />
  </plnk:role>
</plnk:partnerLinkType>
<wsdl:types>
  <xsd:schema elementFormDefault="qualified"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:import namespace="http://xmlns.oracle.com/Enterprise/Tools/
     schemas/PT_WL_GET_INSTANCE_REQ_CONT.v1" schemaLocation="Get
      WorklistEntryStatusRequest.v1.xsd" />
    <xsd:import namespace="http://xmlns.oracle.com/Enterprise/Tools/
      schemas/PT_WL_GET_INSTANCE_RESP_CONT.v1" schemaLocation="
       GetWorklistEntryStatusResponse.v1.xsd" />
    <xsd:import namespace="http://xmlns.oracle.com/schemas/Fault"
     schemaLocation="OperationFault.V1.xsd" />
   </xsd:schema>
  </wsdl:types>
<wsdl:message name="GetWorklistEntryStatusRequest.v1">
  <wsdl:part element="GetWorklistEntryStatusRequest.v1:GetWorklist
   EntryStatusRequest" name="parameter" />
</wsdl:message>
<wsdl:message name="GetWorklistEntryStatusResponse.v1">
  <wsdl:part element="GetWorklistEntryStatusResponse.v1:GetWorklist
   EntryStatusResponse" name="parameter" />
</wsdl:message>
<wsdl:message name="OperationFault.V1">
  <wsdl:part element="OperationFault.V1:OperationFault" name="parameter" />
</wsdl:message>
<wsdl:portType name="PT_WORKLIST_PortType">
<wsdl:operation name="GetWorklistEntryStatus">
  <wsdl:documentation>Get worklist keys and status</wsdl:documentation>
  <wsdl:input message="tns:GetWorklistEntryStatusRequest.v1" name=
   "GetWorklistEntryStatusRequest.v1" />
  <wsdl:output message="tns:GetWorklistEntryStatusResponse.v1" name=
   "GetWorklistEntryStatusResponse.v1" />
  <wsdl:fault message="tns:OperationFault.V1" name="OperationFault.V1" />
 </wsdl:operation>
</wsdl:portType>
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
<wsdl:binding name="PT_WORKLIST_Binding" type="tns:PT_WORKLIST_PortType">
  <soap:binding style="document" transport="http://schemas.xmlsoap.org/
    soap/http" />
  <wsdl:operation name="GetWorklistEntryStatus">
    <soap:operation soapAction="GetWorklistEntryStatus.V1" style=
     "document" />
    <wsp:Policy wsu:Id="UsernameTokenSecurityPolicyPasswordOptional"
      xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
      wss-wssecurity-utility-1.0.xsd">
      <wsp:ExactlyOne>
        <wsp:All>
          <wsse:SecurityToken wsp:Usage="wsp:Required" xmlns:wsse="
           http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
           wssecurity-secext-1.0.xsd">
           <wsse:TokenType>wsse:UserNameToken</wsse:TokenType>
           <Claims>
             <SubjectName MatchType="wsse:Exact" />
             <UsePassword wsp:Usage="wsp:Optional" />
           </Claims>
          </wsse:SecurityToken>
        </wsp:All>
      </wsp:ExactlyOne>
    </wsp:Policy>
   <wsdl:input name="GetWorklistEntryStatusRequest.v1">
     <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      use="literal" />
   </wsdl:input>
   <wsdl:output name="GetWorklistEntryStatusResponse.v1">
     <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      use="literal" />
   </wsdl:output>
   <wsdl:fault name="OperationFault.V1">
     <soap:fault encodingStyle="http://schemas.xmlsoap.org/soap/
       encoding/" name="OperationFault.V1" use="literal" />
   </wsdl:fault>
  </wsdl:operation>
</wsdl:binding>
<wsdl:service name="WorklistServices">
  <wsdl:documentation>Peopletools Worklist</wsdl:documentation>
  <wsdl:port binding="tns:PT_WORKLIST_Binding" name="PT_WORKLIST_Port">
    <soap:address location="http://sbandyop-pc/PSIGW/PeopleSoftService
     ListeningConnector" />
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

Example 3: Asynchronous Request/Response WSDL Document

The following example shows an asynchronous request/response WSDL document provided by the PeopleSoft system:

Copyright © 1988-2007, Oracle. All rights reserved.

```
<?xml version="1.0"?>
<wsdl:definitions name="PT_WORKLIST.1"
targetNamespace="http://xml.namespace.oracle.com/services"
xmlns:CreateWorklistEntryRequest.v1="http://xmlns.oracle.com/Enterprise/
  Tools/schemas/PT_WL_CREATE_REQUEST_CONT.v1"
xmlns:CreateWorklistEntryResponse.v1="http://xmlns.oracle.com/Enterprise/
  Tools/schemas/PT_WL_CREATE_RESPONSE_CONT.v1"
xmlns:plnk="http://schemas.xmlsoap.org/ws/2003/05/partner-link/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http://xml.
  namespace.oracle.com/services" xmlns:wsa="http://schemas.xmlsoap.org/ws/
  2003/03/addressing"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsp="http://schemas.xmlsoap.org/ws/2002/12/policy">
  <wsp:UsagePolicy wsdl:Required="true"/>
  <plnk:partnerLinkType name="PT_WORKLIST_PartnerLinkType">
    <plnk:role name="PT_WORKLIST_Provider">
      <plnk:portType name="tns:PT_WORKLIST_PortType"/>
    </plnk:role>
    <plnk:role name="PT_WORKLIST_Requester">
      <plnk:portType name="tns:PT_WORKLIST_CallbackPortType"/>
    </plnk:role>
  </plnk:partnerLinkType>
  <wsdl:types>
    <xsd:schema elementFormDefault="qualified" xmlns:xsd="http://www.w3.org/
      2001/XMLSchema">
      <xsd:import namespace="http://xmlns.oracle.com/Enterprise/Tools/
        schemas/PT_WL_CREATE_REQUEST_CONT.v1" schemaLocation="CreateWorklist
        EntryRequest.v1.xsd"/>
      <xsd:import namespace="http://xmlns.oracle.com/Enterprise/Tools/
        schemas/PT_WL_CREATE_RESPONSE_CONT.v1" schemaLocation="CreateWorklist
         EntryResponse.v1.xsd"/>
      <xsd:import namespace="http://schemas.xmlsoap.org/ws/2003/03/
        addressing" schemaLocation="http://schemas.xmlsoap.org/ws/2003/
        03/addressing/"/>
    </xsd:schema>
  </wsdl:types>
  <wsdl:message name="CreateWorklistEntryRequest.v1">
    <wsdl:documentation>Create worklist item Request</wsdl:documentation>
    <wsdl:part element="CreateWorklistEntryRequest.v1:CreateWorklist
      EntryRequest" name="parameter"/>
  </wsdl:message>
  <wsdl:message name="CreateWorklistEntryResponse.v1">
    <wsdl:part element="CreateWorklistEntryResponse.v1:CreateWorklist
      EntryResponse" name="parameter"/>
  </wsdl:message>
  <wsdl:message name="InitiateHeader">
    <wsdl:documentation>SOAP Header message for correlating Asynchronous
       callback</wsdl:documentation>
    <wsdl:part element="wsa:MessageID" name="MessageID"/>
    <wsdl:part element="wsa:ReplyTo" name="ReplyTo"/>
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
</wsdl:message>
<wsdl:message name="CallbackHeader">
  <wsdl:documentation>SOAP Header message for callback Asynchronous
     operation
  </wsdl:documentation>
  <wsdl:part element="wsa:RelatesTo" name="RelatesTo"/>
</wsdl:message>
<wsdl:portType name="PT_WORKLIST_PortType">
  <wsdl:operation name="CreateWorklistEntry">
    <wsdl:documentation>Create worklist Entry. This is the Request Operation
      in a Asynchronous Request/Response pair. Callback Operation :
        CreateWorklistEntry_CALLBACK</wsdl:documentation>
    <wsdl:input message="tns:CreateWorklistEntryRequest.v1" name="
       CreateWorklistEntryRequest.v1"/>
  </wsdl:operation>
</wsdl:portType>
<wsdl:portType name="PT_WORKLIST_CallbackPortType">
  <wsdl:operation name="CreateWorklistEntry_CALLBACK">
    <wsdl:documentation>Create worklist Entry - Callback. This is the
       Callback Operation in a Asynchronous Request/Response pair.
    </wsdl:documentation>
    <wsdl:input message="tns:CreateWorklistEntryResponse.v1" name=
      "CreateWorklistEntryResponse.v1"/>
  </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="PT_WORKLIST_Binding" type="tns:PT_WORKLIST_PortType">
  <soap:binding style="document" transport="http://schemas.xmlsoap.
    org/soap/http"/>
  <wsdl:operation name="CreateWorklistEntry">
    <soap:operation soapAction="CreateWorklistEntry.V1" style="document"/>
    <wsp:Policy wsu:Id="UsernameTokenSecurityPolicyPasswordOptional"
      xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
       wssecurity-utility-1.0.xsd">
      <wsp:ExactlyOne>
        <wsp:All>
          <wsse:SecurityToken wsp:Usage="wsp:Required" xmlns:wsse=
            "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
              wssecurity-secext-1.0.xsd">
            <wsse:TokenType>wsse:UserNameToken</wsse:TokenType>
            <Claims>
              <SubjectName MatchType="wsse:Exact"/>
              <UsePassword wsp:Usage="wsp:Optional"/>
            </Claims>
          </wsse:SecurityToken>
        </wsp:All>
      </wsp:ExactlyOne>
    </wsp:Policy>
    <wsdl:input name="CreateWorklistEntryRequest.v1">
      <soap:header encodingStyle="" message="tns:InitiateHeader" part=
        "MessageID" use="literal" wsdl:required="false"/>
```

Copyright © 1988-2007, Oracle. All rights reserved.

```
      <soap:header encodingStyle="" message="tns:InitiateHeader" part=
        "ReplyTo" use="literal" wsdl:required="false"/>
      <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        use="literal"/>
    </wsdl:input>
  </wsdl:operation>
</wsdl:binding>
<wsdl:binding name="PT_WORKLIST_CallbackBinding" type="tns:
   PT_WORKLIST_CallbackPortType">
  <soap:binding style="document" transport="http://schemas.xmlsoap.
    org/soap/http"/>
  <wsdl:operation name="CreateWorklistEntry_CALLBACK">
    <soap:operation soapAction="CreateWorklistEntry_CALLBACK.V1" style=
      "document"/>
    <wsp:Policy wsu:Id="UsernameTokenSecurityPolicyPasswordOptional"
       xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
         wssecurity-utility-1.0.xsd">
      <wsp:ExactlyOne>
        <wsp:All>
          <wsse:SecurityToken wsp:Usage="wsp:Required" xmlns:wsse=
             "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
                wssecurity-secext-1.0.xsd">
            <wsse:TokenType>wsse:UserNameToken</wsse:TokenType>
            <Claims>
              <SubjectName MatchType="wsse:Exact"/>
              <UsePassword wsp:Usage="wsp:Optional"/>
            </Claims>
          </wsse:SecurityToken>
        </wsp:All>
      </wsp:ExactlyOne>
    </wsp:Policy>
    <wsdl:input name="CreateWorklistEntryResponse.v1">
      <soap:header encodingStyle="" message="tns:CallbackHeader"
         part="RelatesTo" use="literal" wsdl:required="true"/>
      <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
         use="literal"/>
    </wsdl:input>
  </wsdl:operation>
</wsdl:binding>
<wsdl:service name="WorklistServices">
  <wsdl:documentation>Peopletools Worklist</wsdl:documentation>
  <wsdl:port binding="tns:PT_WORKLIST_Binding" name="PT_WORKLIST_Port">
    <soap:address location="http://ORACLE_ENDPOINT"/>
  </wsdl:port>
</wsdl:service>
<wsdl:service name="WorklistServices_Callback">
  <wsdl:documentation>Peopletools Worklist - Callback</wsdl:documentation>
  <wsdl:port binding="tns:PT_WORKLIST_CallbackBinding" name=
     "PT_WORKLIST_Callback_Port">
    <soap:address location="http://Client.EndPoint.Set.By.Caller"/>
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

Example 4: Asynchronous One-Way WSDL Document

The following example shows an asynchronous one-way WSDL document provided by the PeopleSoft system:

```
<?xml version="1.0" ?>
<wsdl:definitions name="QEPC_FLO_MSG.1" targetNamespace="http://xmlns.
oracle.com/Enterprise/Tools/services/QEPC_FLO_MSG.1"
   xmlns QEPC_FLO_MSG.VERSION_1="http://xmlns.oracle.com/Enterprise/
     Tools/schemas/QEPC_FLO_MSG.VERSION_1" xmlns:plnk="http://schemas.
   xmlsoap.org/ws/2003/05/partner-link/"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http:
     //xmlns.oracle.com/Enterprise/Tools/services/QEPC_FLO_MSG.1"
   xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
   xmlns:wsp="http://schemas.xmlsoap.org/ws/2002/12/policy">
   <wsp:UsagePolicy wsdl:Required="true" />
   <plnk:partnerLinkType name="QEPC_FLO_MSG_PartnerLinkType">
     <plnk:role name="QEPC_FLO_MSG_Provider">
       <plnk:portType name="tns:QEPC_FLO_MSG_PortType" />
     </plnk:role>
   </plnk:partnerLinkType>
   <wsdl:types>
     <xsd:schema elementFormDefault="qualified"
       targetNamespace="http://xmlns.oracle.com/Enterprise/Tools/schemas/
         QEPC_FLO_MSG.VERSION_1"
       xmlns="http://xmlns.oracle.com/Enterprise/Tools/schemas/
         QEPC_FLO_MSG.VERSION_1"
       xmlns:xsd="http://www.w3.org/2001/XMLSchema">
       <xsd:element name="QEPC_FLO_MSG" type="xsd:string" />
     </xsd:schema>
   </wsdl:types>
   <wsdl:message name="QEPC_FLO_MSG.VERSION_1">
     <wsdl:part element="QEPC_FLO_MSG.VERSION_1:QEPC_FLO_MSG" name=
       "parameter" />
   </wsdl:message>
   <wsdl:portType name="QEPC_FLO_MSG_PortType">
   <wsdl:operation name="QEPC_FLO_MSG">
     <wsdl:documentation>Test</wsdl:documentation>
     <wsdl:input message="tns:QEPC_FLO_MSG.VERSION_1" name="QEPC_FLO_
      MSG.VERSION_1" />
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="QEPC_FLO_MSG_Binding" type="tns:QEPC_FLO_MSG_
    PortType">
    <soap:binding style="document" transport="http://schemas.xmlsoap.
      org/soap/http" />
    <wsdl:operation name="QEPC_FLO_MSG">
      <soap:operation soapAction="QEPC_FLO_MSG.v1" style="document" />
```

Copyright © 1988-2007, Oracle. All rights reserved.

```
        <wsp:Policy wsu:Id="UsernameTokenSecurityPolicyPasswordOptional"
          xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
          wss-wssecurity-utility-1.0.xsd">
            <wsp:ExactlyOne>
              <wsp:All>
                <wsse:SecurityToken wsp:Usage="wsp:Required" xmlns:wsse="
                 http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
                 wssecurity-secext-1.0.xsd">
                   <wsse:TokenType>wsse:UserNameToken</wsse:TokenType>
                   <Claims>
                     <SubjectName MatchType="wsse:Exact" />
                     <UsePassword wsp:Usage="wsp:Optional" />
                   </Claims>
                   </wsse:SecurityToken>
              </wsp:All>
            </wsp:ExactlyOne>
        </wsp:Policy>
        <wsdl:input name="QEPC_FLO_MSG.VERSION_1">
          <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/
           encoding/" use="literal" />
        </wsdl:input>
      </wsdl:operation>
    </wsdl:binding>
    <wsdl:service name="QEPC_FLO_MSG">
      <wsdl:documentation>File Utilities Test</wsdl:documentation>
      <wsdl:port binding="tns:QEPC_FLO_MSG_Binding" name="QEPC_FLO_MSG_Port">
        <soap:address location="http://sbandyop-pc/PSIGW/PeopleSoft
         ServiceListeningConnector" />
      </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

PartnerLinkType Support

A service may play a single or dual role in a partnership with a business process.

In a one-way partnership the service may play a single role of provider, whereas in a two-way partnership the service may play the roles of a provider as well as a requester (for callbacks). This type of partnership is termed by Business Process Execution Language (BPEL) as a PartnerLinkType.

A service may participate in different types of partnerships with a process or another service. In each of these partnerships, the service may play a single or dual role.

PartnerLinkType Structure

To ease the task of process developers consuming PeopleSoft services, a basic PartnerLinkType structure is provided in the PeopleSoft-provided WSDL. Process developers may or may not choose to use this PartnerLinkType structure.

The following table describes details of the PartnerLinkType structure for each service operation type:

 Copyright © 1988-2007, Oracle. All rights reserved.

Operation Type	PartnerLinkType Description
Synchronous	The PartnerLinkType has a single Provider role.
Asynchronous one-way	The PartnerLinkType has a single Provider role.
Asynchronous Request/Response	The PartnerLinkType has two roles for the Provider portType and the Requester Callback portType.

The following sections feature examples of the PartnerLinkType structures the PeopleSoft system generates for each service operation type.

Example 1: Synchronous PartnerLinkType Structure

The following example shows the PartnerLinkType structure that the PeopleSoft system generates for an inbound synchronous service operation:

```
<portType name="HelloWorldSync">
        <operation name="process">
            <input message="client:HelloWorldSyncRequestMessage"/>
            <output message="client:HelloWorldSyncResponseMessage"/>
        </operation>
    </portType>

    <plnk:partnerLinkType name="HelloWorldSyncPLType">
        <plnk:role name="HelloWorldSyncProvider">
              <plnk:portType name="wsdl_target:HelloWorldSync"/>
        </plnk:role>
      </plnk:partnerLinkType>
```

Example 2: Asynchronous One-Way PartnerLinkType Structure

The following example shows the PartnerLinkType structure that the PeopleSoft system generates for an inbound asynchronous one-way service operation.

```
<portType name="UpdateOrderAsync">
        <operation name="UpdateOrder">
            <input message="client:OrderRequestMessage"/>
        </operation>
    </portType>

    <plnk:partnerLinkType name="UpdateOrderAsyncPLType">
        <plnk:role name="UpdateOrderAsyncProvider">
              <plnk:portType name="wsdl_target:UpdateOrderAsync"/>
        </plnk:role>
      </plnk:partnerLinkType>
```

Example 3: Asynchronous Request/Response PartnerLinkType Structure

The following example shows the PartnerLinkType structure that the PeopleSoft system generates for an inbound asynchronous request/response service operation:

Copyright © 1988-2007, Oracle. All rights reserved.

```
<!--
  PortType definition
  -->

    <!-- portType implemented by the QuoteConsumer PeopleSoft service -->
    <portType name="QuoteConsumer">
        <operation name="GetQuote">
            <input message="tns:QuoteConsumerRequestMessage"/>
        </operation>
    </portType>

    <!-- portType implemented by the requester of QuoteConsumer PeopleSoft service
         for asynchronous callback purposes
         -->
    <portType name="QuoteConsumerCallback">
        <operation name="GetQuoteCallback">
            <input message="tns:QuoteConsumerResultMessage"/>
        </operation>
    </portType> <!--
  PartnerLinkType definition
  -->

    <!-- the QuoteConsumer partnerLinkType binds the service and
         requestor portType into an asynchronous conversation.
         -->
    <plnk:partnerLinkType name="QuoteConsumerPLType">
        <plnk:role name="QuoteConsumerProvider">
              <plnk:portType name="wsdl_target:QuoteConsumer"/>
        </plnk:role>
        <plnk:role name="QuoteConsumerRequester">
              <plnk:portType name="wsdl_target:QuoteConsumerCallback"/>
        </plnk:role>
      </plnk:partnerLinkType>
</definitions>
```

WSDL Document Versioning

When the Service System Status in the Services Configuration page is set to *Production* and you attempt to regenerate a WSDL document for a service, PeopleSoft Integration Broker versions and stores the WSDL document in the WSDL repository under the following condition: You have previously generated a WSDL document for the same service and new service operations have been added that you selected to include in the new WSDL document.

In this case, the system appends a version number to the service namespace to enable unique qualification of elements and attributes in the new WSDL version. The new version number is also appended to the WSDL document name. The latest WSDL document version is marked as *Default* in the WSDL repository.

When the Service System Status in the Services Configuration page is set to *Development* and you regenerate a WSDL document for a service, the existing WSDL document is overwritten.

 Copyright © 1988-2007, Oracle. All rights reserved.

Multiple WSDL versions generated from the same service display in a grid on the Services page and include a timestamp and version for each generated WSDL document. Only one of these can be the default version.

See Also

Chapter 14, "Managing Services," Understanding Configuring PeopleSoft Integration Broker for Handling Services, page 277

Chapter 14, "Managing Services," Configuring PeopleSoft Integration Broker for Handling Services, page 277

Chapter 14, "Managing Services," Viewing WSDL Documents Generated for Services, page 283

Prerequisites for Providing Services

The following prerequisites exist for providing services:

- The PeopleSoft system must be configured for handling services. Use the Services Configuration page to define the service namespace, schema namespace, target location and service system status.
- If publishing services to a UDDI repository you must:
  - Have the UDDI server set up, configured and running.
  - Have business entities and categories set up on the UDDI server that you intent to query from the Provide Web Service wizard.
  - Specify the UDDI server and other relevant information within the PeopleSoft system using the Services Configuration-UDDI Configuration page.
- Service operations in services to provide must have any-to-local routing definitions defined.
- There must be a minimum of one service operation associated with the service that you select for which to generate a WSDL document.
- For services that contain service operations with nonrowset-based messages, schemas must exist. The message schema must contain a target namespace. If no target namespace exists an error will occur when the system attempts to generate the WSDL document.

  PeopleSoft automatically generates schema for services that contain operations with rowset-based messages.

See Also

Chapter 14, "Managing Services," Configuring PeopleSoft Integration Broker for Handling Services, page 277

Chapter 14, "Managing Services," Specifying UDDI Repositories in the PeopleSoft System, page 280

Chapter 18, "Managing Routing Definitions," page 331

Chapter 15, "Managing Service Operations," page 291

Chapter 13, "Building Message Schemas," page 265

Copyright © 1988-2007, Oracle. All rights reserved.

Common Elements Used in This Chapter

Description	Description of the service, service operation or WSDL source.
Fault Message	Name of the fault message. Depending on how generated, the name of the fault message can include the version in the following format:*MessageName.Version*.
Operation Typeand**Operation**	Type of service operation. See Chapter 15, "Managing Service Operations," Service Operation Types, page 291.
Request Message	Name and version of the request message in the following format: *MessageName.Version*.
Response Message	Name and version of the response message in the following format: *MessageName.Version*.
Routing External Alias	External Alias from the routing definition for a service operation. Unless overridden, defaults to the format *OperationAlias.Version*.
Select	Check the box to select the WSDL service.
Service	Name of the service that contains the service operations to include in the generated WSDL documents.
Service Operation	Name of the service operations for which to generate WSDL documents.

Providing Services

This section discusses how to use the Provide Web Service wizard to:

- Select services to provide.
- Select service operations.
- View WSDL documents.
- Specify publishing options.
- View the WSDL Generation Log.

Understanding Using the Provide Web Service Wizard

The Provide Web Service component (IB_WSDLEXP_SRCH) features a wizard you can use to provide web services. You can publish WSDL document to the WSDL repository in the PeopleSoft system or external UDDI repositories.

After you generate a WSDL document, the Provide Web Service wizard display a WSDL URL for each document you generated. This enables you to access the WSDL document content using the WSDL URL. In addition, you can modify the URL to access the WSDL document content using a WSIL URL.

 Copyright © 1988-2007, Oracle. All rights reserved.

Note. For a service to be available to provide, an any-to-local routing must exist for the service. In addition, there must be a minimum of one service operations associated with the service.

You can use the Provide Web Service wizard to select one or more services for which to generate WSDL documents. A separate WSDL document is generated for each service.

See Also

Chapter 23, "Providing Services," Prerequisites for Providing Services, page 521

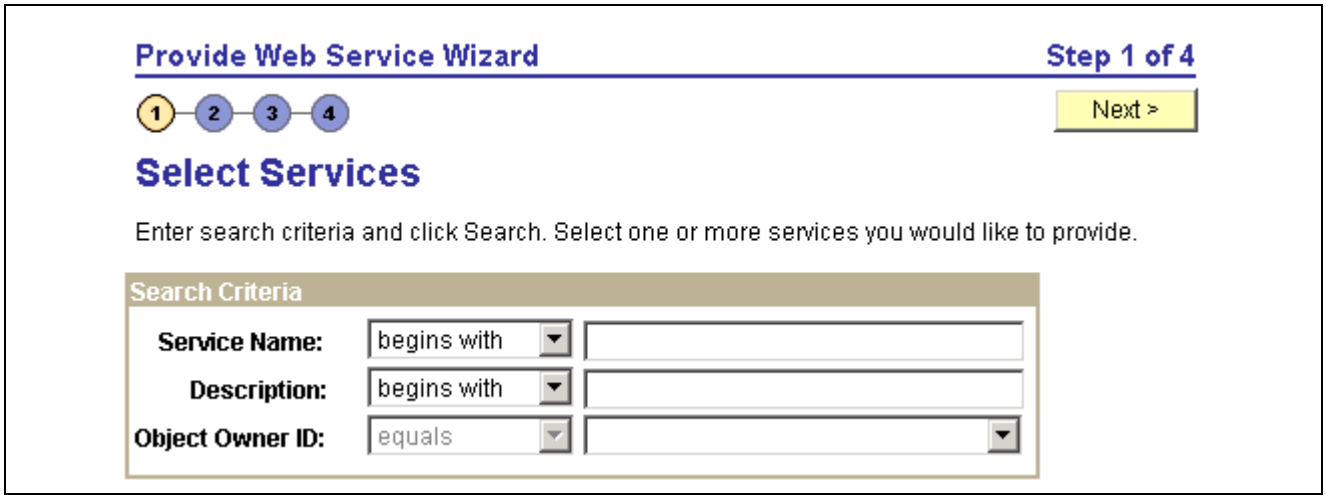
Pages Used in Using the Provide Web Service Wizard

Page Name	Object Name	Navigation	Usage
Select Services page	IB_WSDLEXP_SRCH	Select PeopleTools, Integration Broker, Web Services, Provide Web Service.	Select services that contain the service operations to include in WSDL documents.
Select Service Operations page	IB_WSDLEXP_OPER	On the Select Services page, click the Next button. You can also access this page from a service definition in the Services component. Select PeopleTools, Integration Broker, Integration Setup, Services. Click the Provide Web Service link. For the Provide Web Service link to appear on the Services page, an any-to-local routing must be defined for the service.	Select the service operations contained in a service to include in the generated WSDL document.
View WSDL	IB_WSDLEXP_PVIEW	On the Select Service Operations page, click the Next button.	Use the page to view WSDL before generating a WSDL document.
WSDL Viewer	IB_WSDEXPPVIEW_SEC	On the View WSDL page, click the View WSDL link.	Use the page to: • View a WSDL document before generating it. • Copy and paste a WSDL document to file before generating it.
Specify Publishing Options	IB_WSDLEXP_LOC	On the View WSDL page, click the Next button.	Select the location of where to publish WSDL documents.
Results page	IB_WSDLEXP_RSLTS	On the Specify Publishing Options page, click the Finish button.	Generate WSDL documents for selected service operations and view the WSDL generation log.

Copyright © 1988-2007, Oracle. All rights reserved.

Step 1: Select Services to Provide

Use the Provide Web Service-Select Services page to search for and select the services that contain the service operations to include the WSDL documents that you generate. The following example shows the page:



Provide Web Service - Select Services to Provide page

You can search by the full or partial service name and service description. You can also search by object owner ID, if one is defined for the service. You can enter one or more of these criteria when performing your search.

To select services to provide:

1. Access the Provide Web Service–Select Services to Provide page.
2. Enter search criteria for the services to provide by performing one or more of the following:
   - In the Service Name field, enter a full or partial service name.
   - In the Description field, enter the full or partial description of the service.
   - From the Object Owner ID dropdown list box, select the object owner of the service to provide.
   - Select no search criteria to retrieve a list of all services in the database for which any-to-local routing definitions have been generated.
3. Click the Search button.

   A Services grid appears that contains the search results.

   The search results only list services which have at least one service operation with an any-to-local routing.
4. Check the box next to each name of the services to provide.

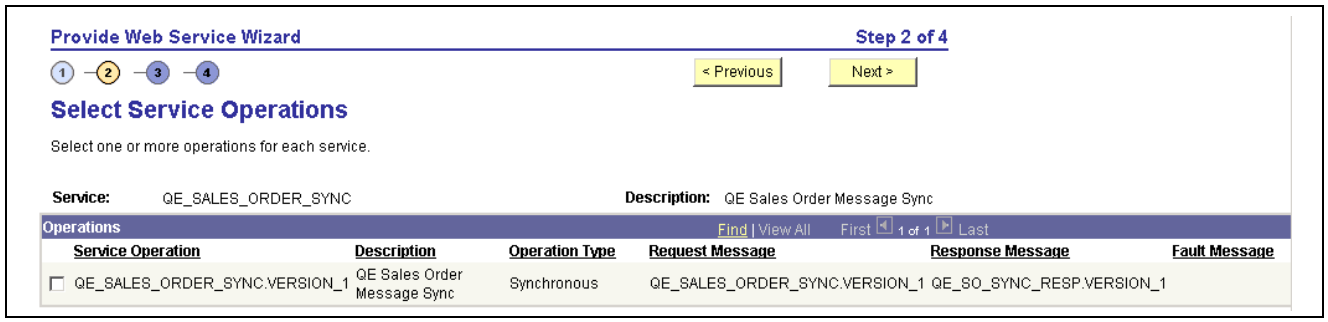
   To clear a selection, check the box again.
5. Click the Next button to proceed to the next step in the wizard.

The next section discusses the next step to providing a service, selecting the operations of the service to provide.

Step 2: Select Service Operations

The following graphic shows the Provide Web Service–Select Service Operations page:

Copyright © 1988-2007, Oracle. All rights reserved.



Provide Web Service - Select Service Operations

Use the page to select the service operations from each service that you selected in the previous step to include in the WSDL document.

To select service operations to include in a WSDL document:

1. Check the box next to each service operation to include.

   To clear a selection, check the box again.

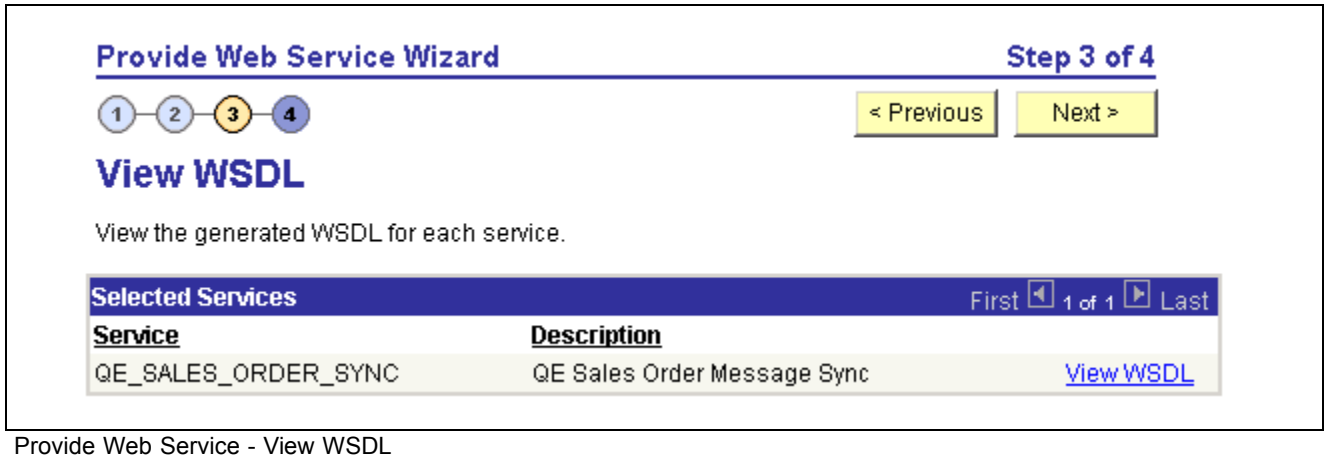
2. Click the Next button to proceed to the next step in the wizard.

The next step to providing WSDL documents is previewing the WSDL document that will be provided.

Step 3: View WSDL Documents

After you select the service operations to include in a WSDL document, you can preview the WSDL before actually publishing it.

The following graphic shows the Provide Web Service -View WSDL page. Use the page to preview a WSDL document before you actually generate it.

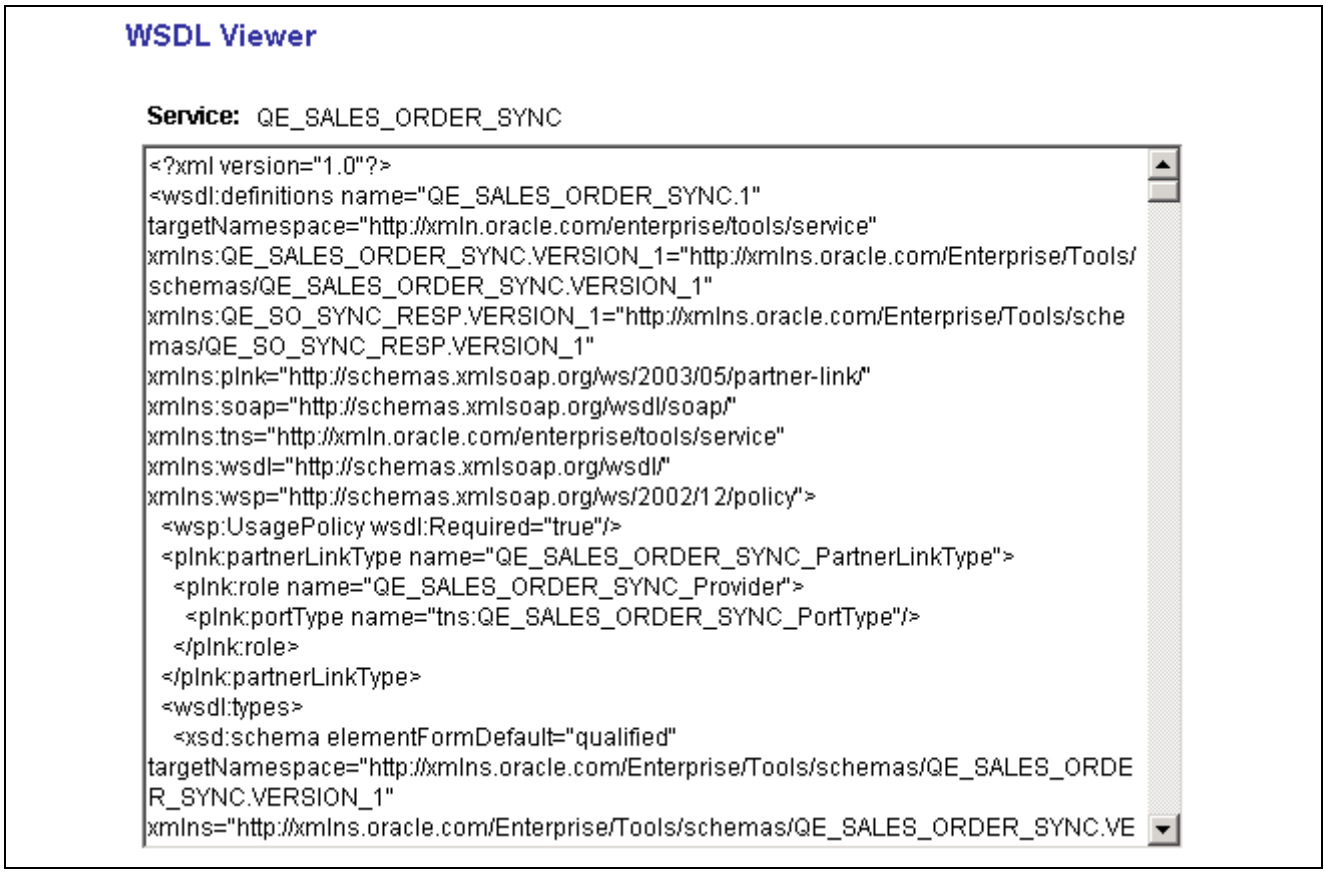


Provide Web Service - View WSDL

Each service for which a WSDL document will be generated is listed. Click the View WSDL link to view the WSDL document for each service that you have selected.

When you click the View WSDL link, the WSDL displays in the WSDL Viewer page. The following example shows the WSDL document for the *QE_SALES_ORDER_SYNC* service in the WSDL Viewer page.

Copyright © 1988-2007, Oracle. All rights reserved.



Provide Web Service-WSDL Viewer

To preview WSDL documents:

1. Click the View WSDL link for a service.

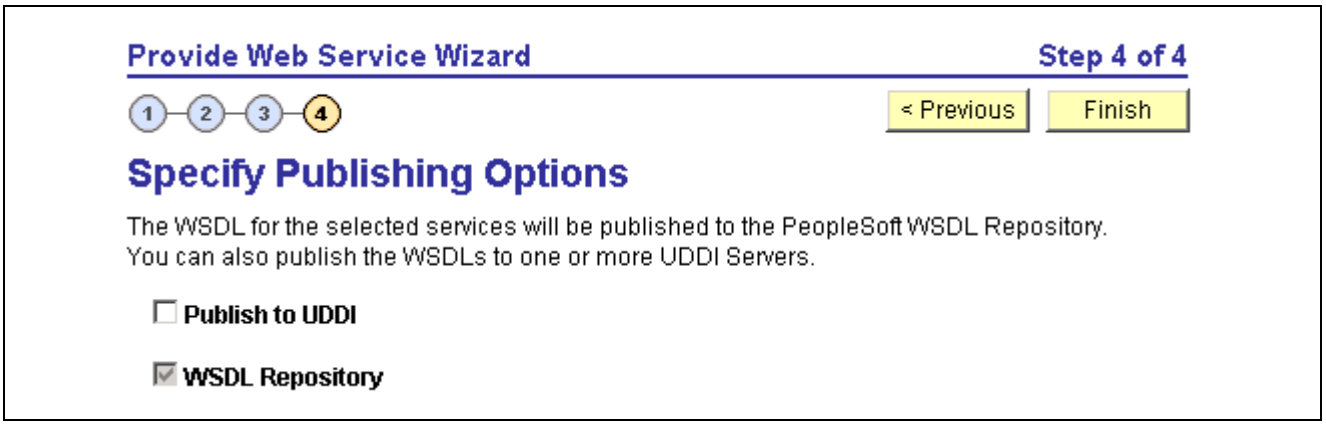
   The WSDL document for the service appears in the WSDL Viewer.
2. Click the Return button to return to the View WSDL page.
3. Click the Next button to proceed to the next step in the wizard.

The next section discusses the next step to providing a service, selecting the location of where to publish the WSDL documents.

Step 4: Specify Publishing Options

After you preview the WSDL, use the Specify Publishing Options page to specify the publish location of the generated WSDL documents. The following graphic shows the page:

Copyright © 1988-2007, Oracle. All rights reserved.



Provide Web Service - Specify Publishing Options page

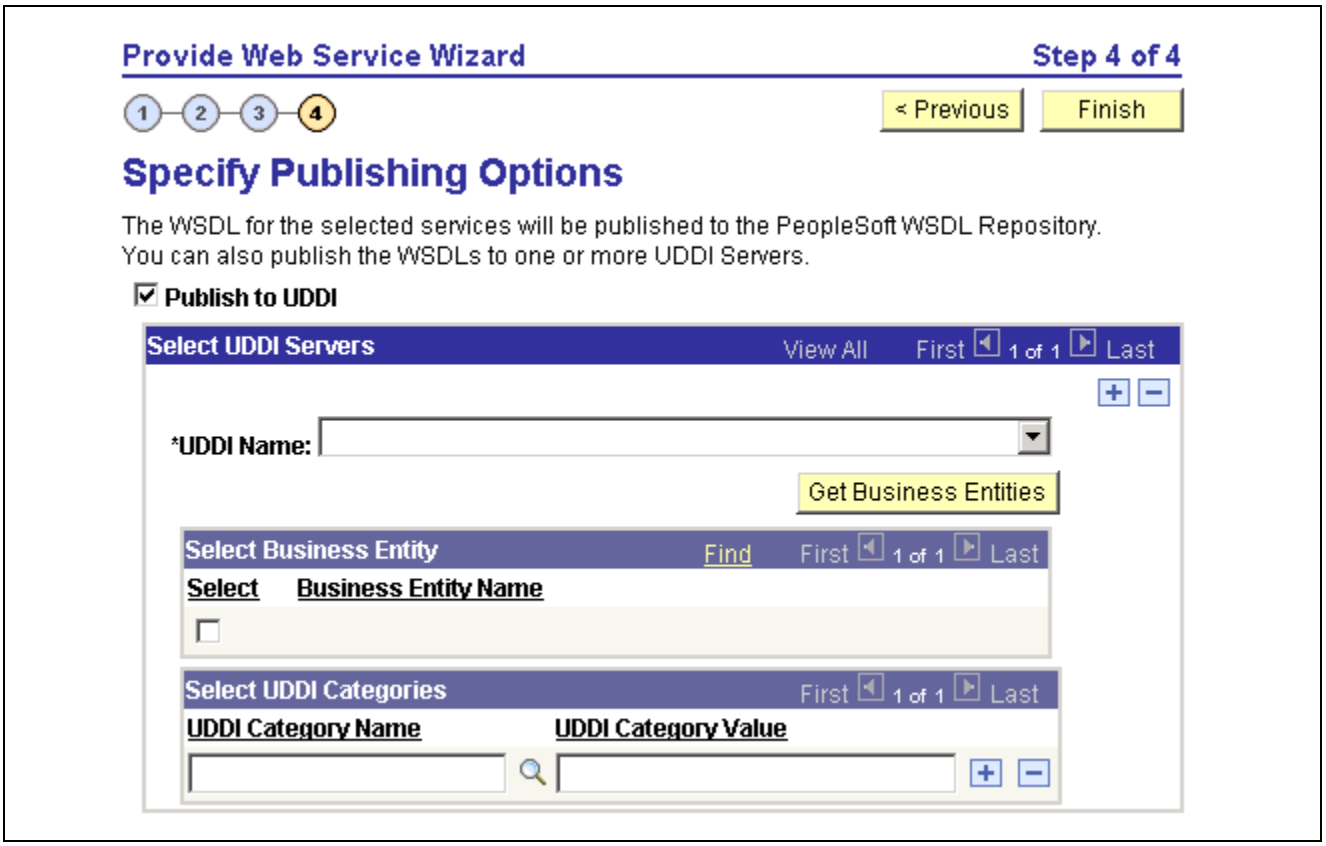
By default the system publishes all WSDL documents to the PeopleSoft WSDL repository.

Select the Publish to UDDI check box to publish the WSDL to a UDDI repository in addition to the PeopleSoft WSDL repository.

Providing WSDL Documents to UDDI Repositories

Before providing a WSDL document to a UDDI repository, you must configure the UDDI repository in the PeopleSoft system.

When you select the Publish to UDDI check box, the Select UDDI Servers box appears where you specify the UDDI repository to which to publish the WSDL. The following graphic shows the Select UDDI Servers section:



Select UDDI Servers section

Copyright © 1988-2007, Oracle. All rights reserved.

To provide a WSDL document to a UDDI repository:

1. From the UDDI Name dropdown list box, select the UDDI server to which you are publishing the WSDL.
2. Click the Get Bus. Entities button.

   The Select Business Entity section lists the business entities that are available to select for the UDDI server.
3. Check the box next to each business entity name to include.
4. Click the UDDI Category Name lookup button to display a list of UDDI categories and select a UDDI category. Click the OK button.
5. In the Category Value field, enter a value for the category.
6. To add additional categories, in the Select UDDI Categories section, click the plus button to add a row and repeat step 5 and step 6.
7. Click the Finish button.

   The Results page appears and displays the WSDL generation log.

Step 5: View the WSDL Generation Log

Use the Results page shown in the following example to view the WSDL Generation Log:

Provide Web Service-Confirm Results page

The WSDL Generation Log provides the name of the services and URL for each WSDL document generated.

You can cut and paste the URL into a browser to access the WSDL document. You can also access the WSDL document using the WSDL repository.

To provide another service, click the Provide Another Service button and return to step 1 of the wizard.

 Copyright © 1988-2007, Oracle. All rights reserved.

You can also click the Generate SOAP Template button to access the Generate SOAP utility to generate SOAP message templates for request messages, response messages and fault messages found in the WSDL document. You can then use the templates to test SOAP messages in the Handler Tester, Transformation Test Tool and Send Master utilities.

See Also

Enterprise PeopleTools 8.49 PeopleBook: PeopleSoft Integration Testing Utilities and Tools, "Generating SOAP Template"

Accessing Generated WSDL Documents

This section discusses how to:

- Use WSDL URLs to access generated WSDL documents.
- Use WSIL URLs to access generated WSDL documents.
- Use the WSDL repository to access generated WSDL documents.

Pages Used to Access Generated WSDL Documents

Page Name	Object Name	Navigation	Usage
Results page	IB_SERVICEWSDL_SEC	On the Specify Publishing Options page of the Provide Web Service wizard, click the Finish button.	Access WSDL URLs for generated WSDL documents. Use the WSDL URL as-is to generate the WSDL document, or modify the URL to access WSDL document content using a WSIL URL.
WSDL Repository page	IB_SERVICEDEFN_SEC	Select PeopleTools, Integration Broker, Integration Setup, Services. The Service page appears. Click the View WSDL link.	View a list of generated WSDL documents that exist for a service. Click the View WSDL link for a service operation to access the WSDL document.

Using WSDL URLs To Access Generated WSDL Documents

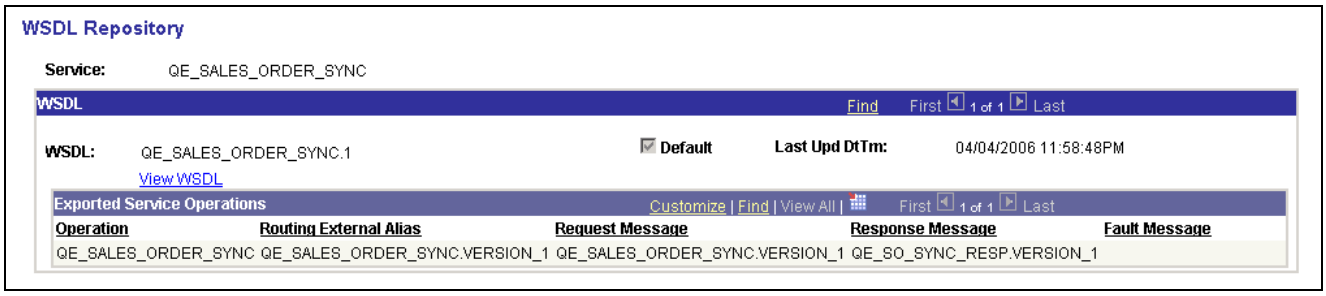
The last page of the Provide Web Service wizard, the Results page, displays a WSDL generation log. The text box contains WSDL URLs for each WSDL document you generated.

To access the WSDL, copy a WSDL URL from the WSDL generation log and paste it into a browser of your choice to access the WSDL document.

Using the WSDL Repository to Access Generated WSDL Documents

The following example shows the WSDL Repository page:

Copyright © 1988-2007, Oracle. All rights reserved.



WSDL Repository page for the QE_SALES_ORDER_SYNC service

The WSDL Repository page lists the WSDL documents that exist for a service.

The previous example shows that one WSDL document, *QE_SALES_ORDER_SYNC.1*, exists for the *QE_SALES_ORDER_SYNC* service.

The selected and disabledDefault check box indicates the default WSDL document for the service. The WSDL document last generated is the default version. .

Note. Default WSDL documents for a service are used only when the service system status is *Production*.

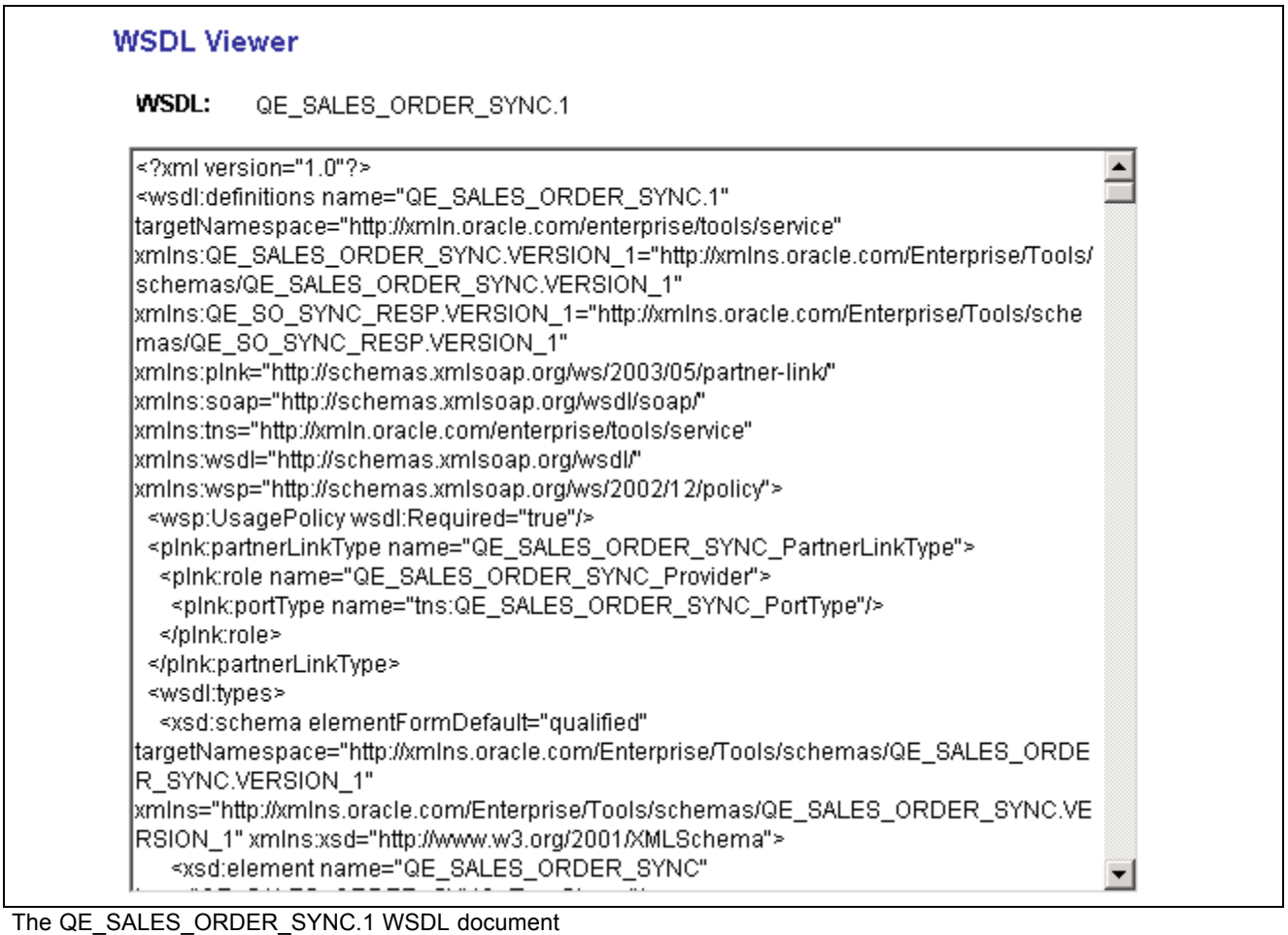
To change the default version, you must provide the service again, and choose a different set of service operations to include in the new default version.

Descriptions of the other fields displayed on this page are discussed at the beginning of this chapter.

See Chapter 23, “Providing Services,” Common Elements Used in This Chapter, page 522.

To access the WSDL document, click the View WSDL link. the WSDL Viewer page appears and displays the content of the WSDL document. The following example shows the WSDL Viewer displaying the *QE_SALES_ORDER_SYNC.1* WSDL document:

 Copyright © 1988-2007, Oracle. All rights reserved.



The QE_SALES_ORDER_SYNC.1 WSDL document

Click the Return button to return to the WSDL Repository page.

Deleting WSDL Documents

This section discusses how to delete WSDL documents.

Understanding Deleting WSDL Documents

The service system status affects the ability to delete WSDL documents.

See Chapter 14, “Managing Services,” Understanding Configuring PeopleSoft Integration Broker for Handling Services, page 277 and Chapter 14, “Managing Services,” Setting Service Configuration Properties, page 279.

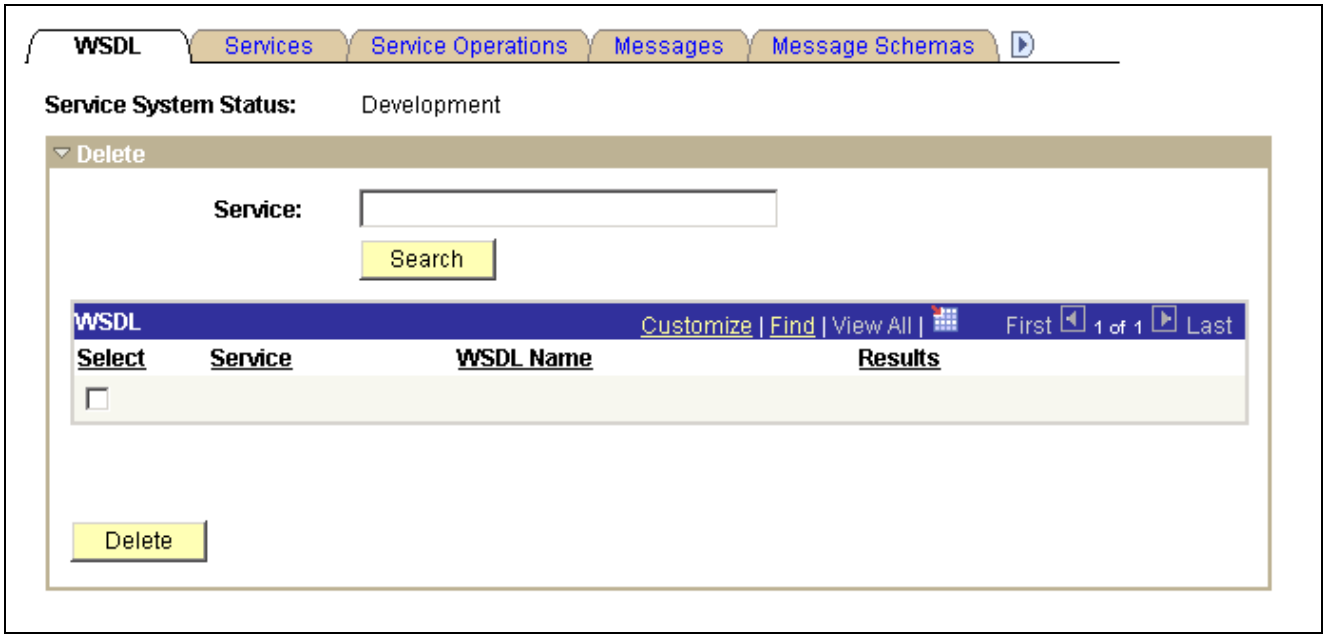
Copyright © 1988-2007, Oracle. All rights reserved.

Page Used to Delete WSDL Documents

Page Name	Object Name	Navigation	Usage
Service Administration-WSDL page	IB_HOME_PAGE7	Select PeopleTools, Integration Broker, Service Utilities, Service Administration. Click the WSDL tab.	Delete WSDL documents.

Deleting a WSDL Document

Use the Service Administration-WSDL page to delete WSDL documents generated for a service. The following example shows the page:



Service Administration-WSDL page

To delete a WSDL document

1. Access the Services Administration - WSDL page.
2. Click the arrow next to the Delete section header to expand the section.
3. In the Service field, enter the name of the service to delete, and click the Search button.

   Search results appear in the results grid.
4. In the results grid, select the check box next to the WSDL document to delete.
5. Click the Delete button.

Copyright © 1988-2007, Oracle. All rights reserved.

CHAPTER 24

Consuming Services

This chapter discuses how to:

- Set the PS_FILEDIR environment variable for consuming WSDL from files.
- Consume services.
- Access integration metadata for consumed services.

Understanding Consuming Services

PeopleSoft Integration Broker can consume external services by way of consuming WSDL documents and generates PeopleSoft integration metadata, so that PeopleSoft applications can invoke outbound synchronous and outbound asynchronous services.

PeopleSoft Integration Broker features a Consume Web Service wizard that steps you through the task of consuming WSDL documents.

Understanding the Consume Web Service Wizard

This section discusses the Consume Web Service wizard.

Consume Web Service Wizard Features

The Consume Web Service wizard supports:

- WS-interoperability standards for WSDL.
- Consumption of WSDL version 1.1 documents.
- Consuming WSDL with SOAP, HTTP–GET/POST bindings.
- Nested URIs to resolve WSDL fragments.

Operation Types Supported

You can consume WSDL documents for the following service operation types:

- Synchronous.
- Asynchronous one-way.
- Asynchronous request/response.

Copyright © 1988-2007, Oracle. All rights reserved.

Sources for Consuming WSDL Document

You can use the Consume Web Service wizard to consume WSDL documents from the following sources:

- UDDI repositories.
- WSDL URL.
- WSIL registries.
- File.
- Legacy PeopleSoft WSDL documents.

Integration Metadata Created by the Consume Web Service Wizard

The Consume Web Service wizard creates the following integration metadata in the PeopleSoft system from the consumed WSDL documents:

Note. The internal name is the name that the PeopleSoft system assigns to the metadata and is the definition name that appears in the PeopleSoft system.

Metadata	Internal Name	Version	Comments
Service definitions	Same name as the <service> element name in the WSDL document that the PeopleSoft system consumed.	Version one, denoted as: *.V1*	NA
Service operation definitions	Same name as the <service operation> element name in the WSDL document that the PeopleSoft system consumed.	Version one, denoted as: *.V1*	NA

Copyright © 1988-2007, Oracle. All rights reserved.

Metadata	Internal Name	Version	Comments
Message definitions	System-generated name in the format *M<unique number>*. For example: *M120508438*.	Version one, denoted as: *.V1*	Includes request messages, response messages. and fault messages, as appropriate. Messages can be unstructured or containers. You can rename the system-generated message names in the wizard using more meaningful names. The consume process also generates schemas for each message. All schemas are unstructured.
Routing definitions	System-generated name in the format *~IMPORTED~<unique number>*. For example: *~IMPORTED~14857*.	Version one, denoted as: *.V1*	System creates a point-to-point routing.

Multiple Fault Messages

If a WSDL operation has multiple faults, PeopleSoft Integration Broker creates a part message for each fault in the WSDL operation. The system then creates a container message and places the fault part messages in the container. The container message is assigned as a fault message to the created service operation.

Multiple Root Elements in Message Schemas

In a WSDL document, the schema defined in the <types> section may have multiple root elements, corresponding to multiple messages used by one or more operations. When the PeopleSoft system consumes such a WSDL document, the entire message schema contained in the WSDL document gets associated with each of the service operation messages when the PeopleSoft system generates the integration metadata.

Use the element name that appears in the Comments text box of the message definition to construct the XML data for the correct schema fragment in the message.

Delivered Queues and Nodes

PeopleSoft delivers a queue, *WSDL_QUEUE*, and a node, *WSDL_NODE*, that the Consume Web Service wizard uses as defaults.

You may uses these objects or select other existing queues or nodes.

Binding Style of Consumed WSDL Documents

The binding style of consumed WSDL documents appears in the service operation definition for the consumed service. The Default Service Operation Version section of the Service Operations page features a Comments box. The binding style appears in that box.

Copyright © 1988-2007, Oracle. All rights reserved.

Working with Asynchronous Request/Response Service Operations

If a WSDL document has two port types with a single input message in each operation, the Consume Web Service wizard displays step where you can convert a pair of asynchronous one-way operations to a single asynchronous request/response operation. In this step you can special the request and callback service operations and convert the operation.

Prerequisites for Consuming Services

To consume services you must set properties in the Service Configuration component as follows:

- Use the Services Configuration page to define the service namespace, schema namespace, target location and service system status.
- If consuming services from a UDDI repository, you must first specify the UDDI server and other relevant information within the PeopleSoft system using the Services Configuration-UDDI Configuration page.
- To consume WSDL from a file, you must set the PS_FILEDIR environment variable on your machine.

Common Elements Used in This Chapter

Description	Description of the WSDL source.
Endpoint	According to the W3C, “An association between a binding and a network address, specified by a URI, that may be used to communicate with an instance of a service.” A URI that accepts messages containing document-oriented or procedure-oriented information.
Internal Message	Name of the consumed request message, response message or fault message in the PeopleSoft system
Internal Operation	Name of the consumed service operation in the PeopleSoft system
Internal Service	Name of the consumed service in the PeopleSoft system
Next	Click the button to advance to the next step in the wizard.
Operation Type/Operation	Type of service operation. See Chapter 15, “Managing Service Operations,” Service Operation Types, page 291.
Previous	Click the button to go back to the previous step in the wizard.
Select	Check the box to select the WSDL service or WSDL operation on which to perform an action.
View WSDL	Click the link to view the WSDL document for a service from the WSDL source.

 Copyright © 1988-2007, Oracle. All rights reserved.

WSDL Port	According to the W3C: • "A port indicates a specific location for accessing a service using a specific protocol and data format." • "The network address of an endpoint and the binding to which it adheres."
WSDL Fault Message	Name of the fault message specified in the WSDL document that the PeopleSoft system is consuming.
WSDL Request Message	Name of the request message specified in the WSDL document that the PeopleSoft system is consuming.
WSDL Response Message	Name of the response message specified in the WSDL document that the PeopleSoft system is consuming.
WSDL Service	The name of the external service in the WSDL document that the PeopleSoft system is consuming.
WSDL Source	Indicates the source of the service that the PeopleSoft system is consuming.
WSDL URL	Displays the WSDL URL, WSIL URL, File name or UDDI server name of the WSDL source.

See Also

http://www.w3.org/TR/wsdl

Setting the PS_FILEDIR Environment Variable for Consuming WSDL from Files

This section discusses how to:

- Set the PS_FILEDIR environment variable in Microsoft Windows environments.
- Set the PS_FILEDIR environment variable in Unix environments.

Understanding Setting the PS_FILEDIR Environment Variable

Before you can use PeopleSoft Integration Broker to consume WSDL from a file, you must set the PS_FILEDIR environment variable on your machine. If you do not set this variable and attempt to consume WSDL from a file, you will receive an error that the WSDL cannot be parsed.

Note. You must set this variable only if you will be consuming WSDL from a file.

Setting PS_FILEDIR in Microsoft Windows Environments

To set the PS_FILEDIR environment variable in Microsoft Windows environments:

1. Close any open DOS windows.
2. On your desktop, right-click the My Computer icon and click Properties.

   The System Properties dialog appears.

Copyright © 1988-2007, Oracle. All rights reserved.

3. Click the Advanced tab.
4. In the Environment Variables section, click the Environment Variables button.

   The Environment Variables dialog box appears.
5. In the User variables for <user name> section, click New.

   A New User Variable dialog box displays.
6. In the Variable Name field enter *PS_FILEDIR*.
7. In the Variable Value field, enter *c:\<path>*.

   The path you specify is the location from where the system will upload files.
8. Click the OK button to exit the Environment Variables dialog box.
9. Click the OK button again to exit the System Properties dialog box.

Setting PS_FILEDIR in Unix Environments

To set the PS_FILEDIR variable in Unix use one of the following commands as appropriate for your Unix environment:

- export PS_FILEDIR = <PS_HOME>/file
- setenv PS_FILEDIR = <PS_HOME>/file

The path you specify in either command is the location from where the system will upload files.

Using the Consume Web Service Wizard

This section discusses how to use the Consume Web Service wizard to:

- Select the WSDL source.
- Select a service.
- Select a service port.
- Select service operations.
- Convert asynchronous operations.
- Rename operation messages.
- Select a queue for asynchronous operations.
- Select the receiver node.
- Confirm and view results.

 Copyright © 1988-2007, Oracle. All rights reserved.

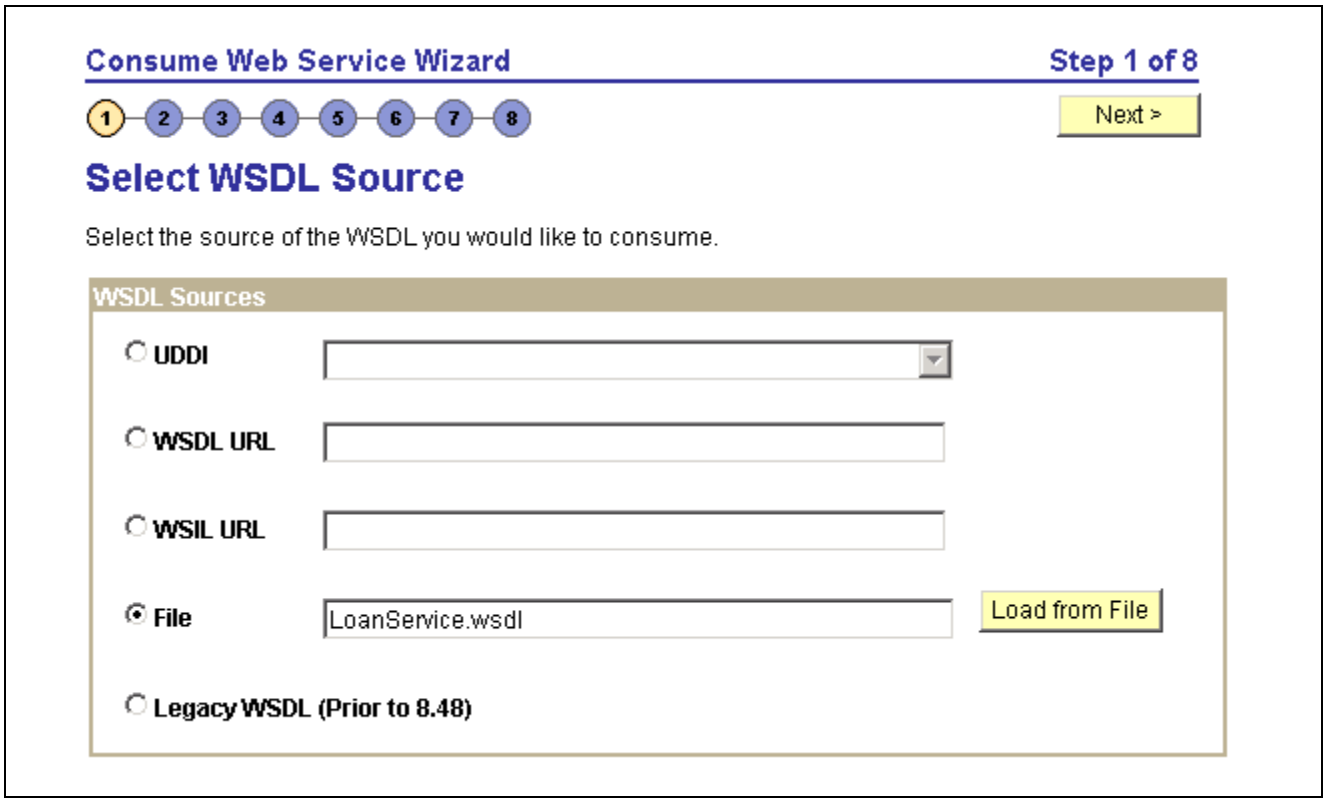
Pages Used to Consume Services

Page Name	Object Name	Navigation	Usage
Select WSDL Source page	IB_WSDL_IMP_LOC	Select PeopleTools, Integration Broker, Web Services, Consume Web Service.	Select the source from which to consume a WSDL document.
Select Service page	IB_WSDL_IMP_SVC	From the Select WSDL Source page, click the Next button.	Select the service to consume.
Select Service Ports page	IB_WSDL_IMP_SVC2	From the Select Service page, click the Next button.	This page appears if a service you select has more than one port. Use the page to select the service ports to use.
Select Service Operations page	IB_WSDL_IMP_OPR	From the Select Service page or from the Select Service Ports page, click the Next button.	Select service operations from the selected services to consume into the PeopleSoft system
Convert Asynchronous Operations page	IB_WSDL_IMP_ASYNOP	From the Select Service Operations page, click the Next button.	The page appears when the system detects that you are consuming two asynchronous one-way operations. Use the page to convert the two operations into a single asynchronous request/response operation type.
Rename Operation Messages page	IB_WSDL_IMP_MSGS	From the Select Service Operations page, click the Next button.	Rename messages generated by the system using more meaningful names.
Select a Queue for Asynchronous Operations page	IB_WSDL_IMP_QUEUE	From the Rename Operation Messages page, click the Next button.	This page appears only when working with asynchronous operation types. Use the page to specify PeopleSoft queues for asynchronous service operations.
Select the Receiver Node page	IB_WSDL_IMP_NODE	From the Select a Queue for Asynchronous Operations page, click the Next button.	Use the page to select the PeopleSoft node that will receive the service operation.
Confirm Results	IB_WSDL_IMP_RSLTS	From the Select the Receiver Node page, click the Finish button.	Use this page to view the results of consuming the service.

Step 1: Select WSDL Source

Use the Consume Web Service-Select WSDL Source page to select the source for consuming a WSDL document. The following example shows the page:

Copyright © 1988-2007, Oracle. All rights reserved.



Select WSDL Source page

To select the WSDL source:

1. Select the radio button next to one of the following values and enter the information specified:

UDDI

To consume a WSDL document from a UDDI repository, select theUDDI radio button.

After you select the radio button, select the UDDI repository from the dropdown list box.

To use this option, you must first specify the UDDI repository in the PeopleSoft system.

See Chapter 24, "Consuming Services," Prerequisites for Consuming Services, page 536.

WSDL URL

To consume a WSDL document from a WSDL URL, select the WSDL URL radio button.

After you select the radio button, enter the URL in the WSDL URL field.

WSIL URL

To consume a WSDL document from a WSIL URL, select the WSIL URL radio button.

After you select the radio button, enter a URL in the WSIL URL field.

File

To consume a WSDL document from a file, perform one of the following actions:

- In the File field, enter the path and file name. For example: *c:\temp\sample.wsdl*.

Copyright © 1988-2007, Oracle. All rights reserved.

- Browse for the file location and name.
  1. Click the Load from File button.

     A file upload page appears.
  2. Click the Browse button to search for and select the file location and name.
  3. Click the Upload button.

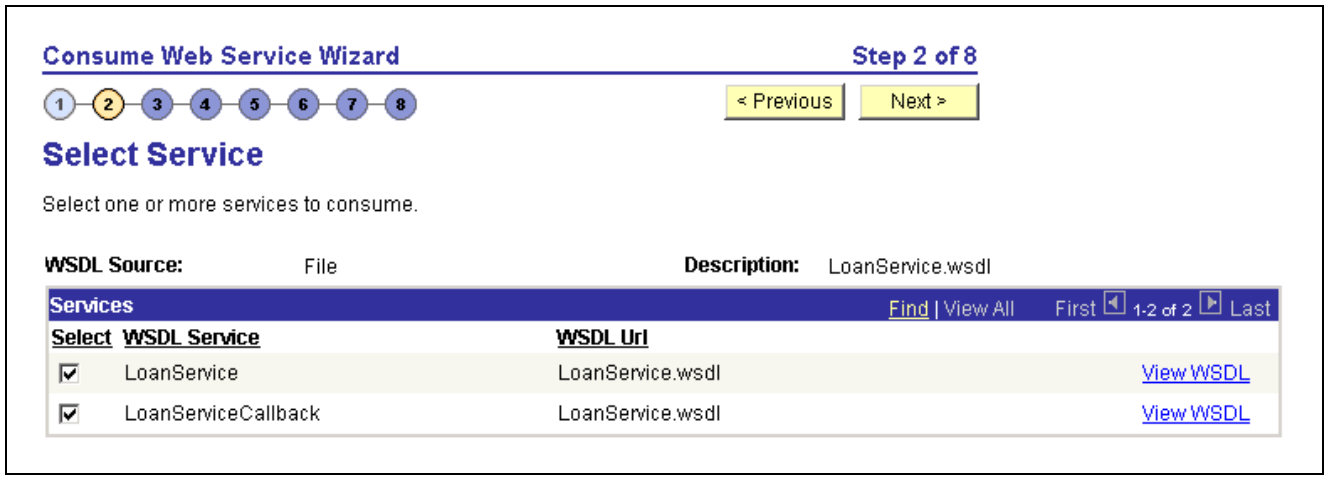
     The Select WSDL Source page appears with the file location and name populated in the File field.

Legacy WSDL (Prior to 8.48) Select this option to consume a PeopleSoft WSDL document generated from releases prior to PeopleTools 8.48.

4. Click the Next button to proceed to the next step in the wizard.

Step 2: Select Service

Use the Consume Web Service-Select Service page to select the services to consume. The following example shows the page:



Consume Web Service-Select Service page

Before selecting services to consume, you can click the View WSDL link to view the WSDL for each service. The WSDL document opens in a browser. Close the browser when you have finished viewing the WSDL document.

WSDL documents that the PeopleSoft system consumes from pre-PeopleTools 8.48 systems display in an edit box.

To select services to consume:

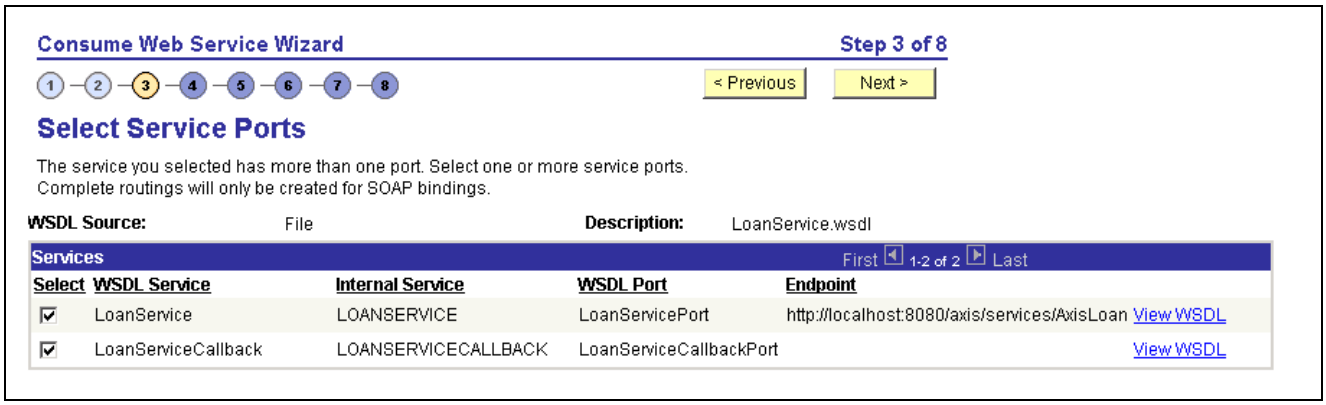
1. Check the box next to each service to consume.

   To clear a selection, check the box again.
2. Click the Next button to proceed to the next step in the wizard.

Step 3: Select Service Ports

If a service you select has more than one port, the Consume Web Service–Select Service Ports page appears. The following example shows the page:

Copyright © 1988-2007, Oracle. All rights reserved.



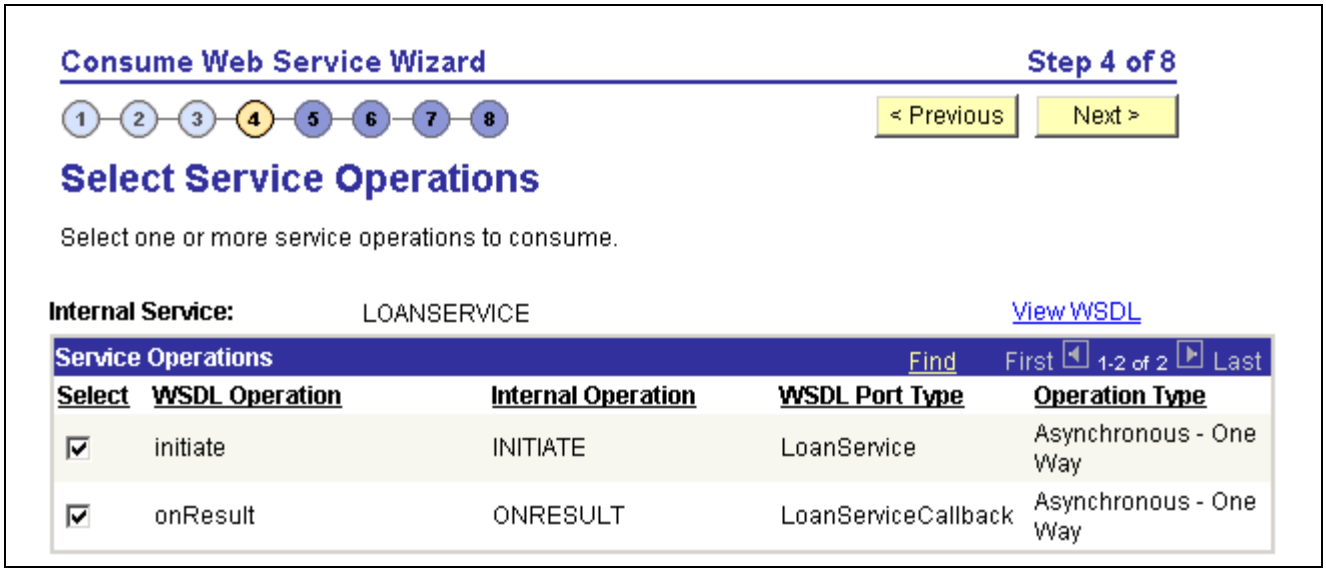
Consume Web Service-Select Service Ports page

Multiple port options only appear when you are working with asynchronous request/response operations in a WSDL document or the service has multiple bindings. Typically, when working with this operation type, you select both options.

To select service ports, in the Select column, check the box next to each service.

Step 4: Select Service Operations

Use the Consume Web Service-Select Service Operations page to select the operations in the selected service to consume. The following example shows the page:



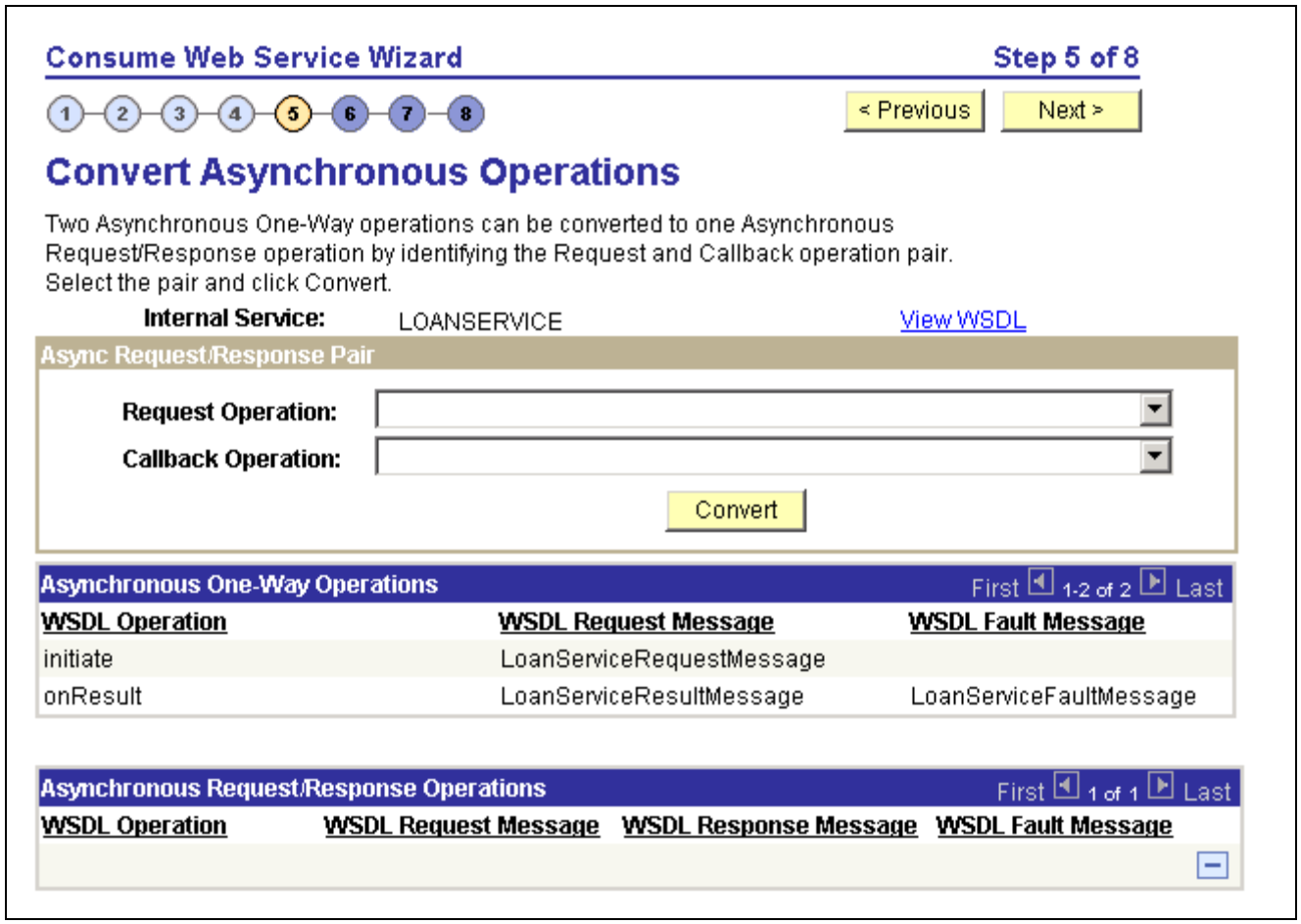
Consume Web Service-Select Service Operations page

To select service operations to consume, in the Select column, check the box next to each service operation to consume.

Step 5: Convert Asynchronous Operations

The Consume Web Service-Convert Asynchronous Operations page is shown in the following example:

Copyright © 1988-2007, Oracle. All rights reserved.



Consume Web Service-Convert Asynchronous Operations

The page appears when the system detects that you are consuming at least two asynchronous one-way operations. The two asynchronous one-way operations appear in the Asynchronous One-Way Operations section on the page.

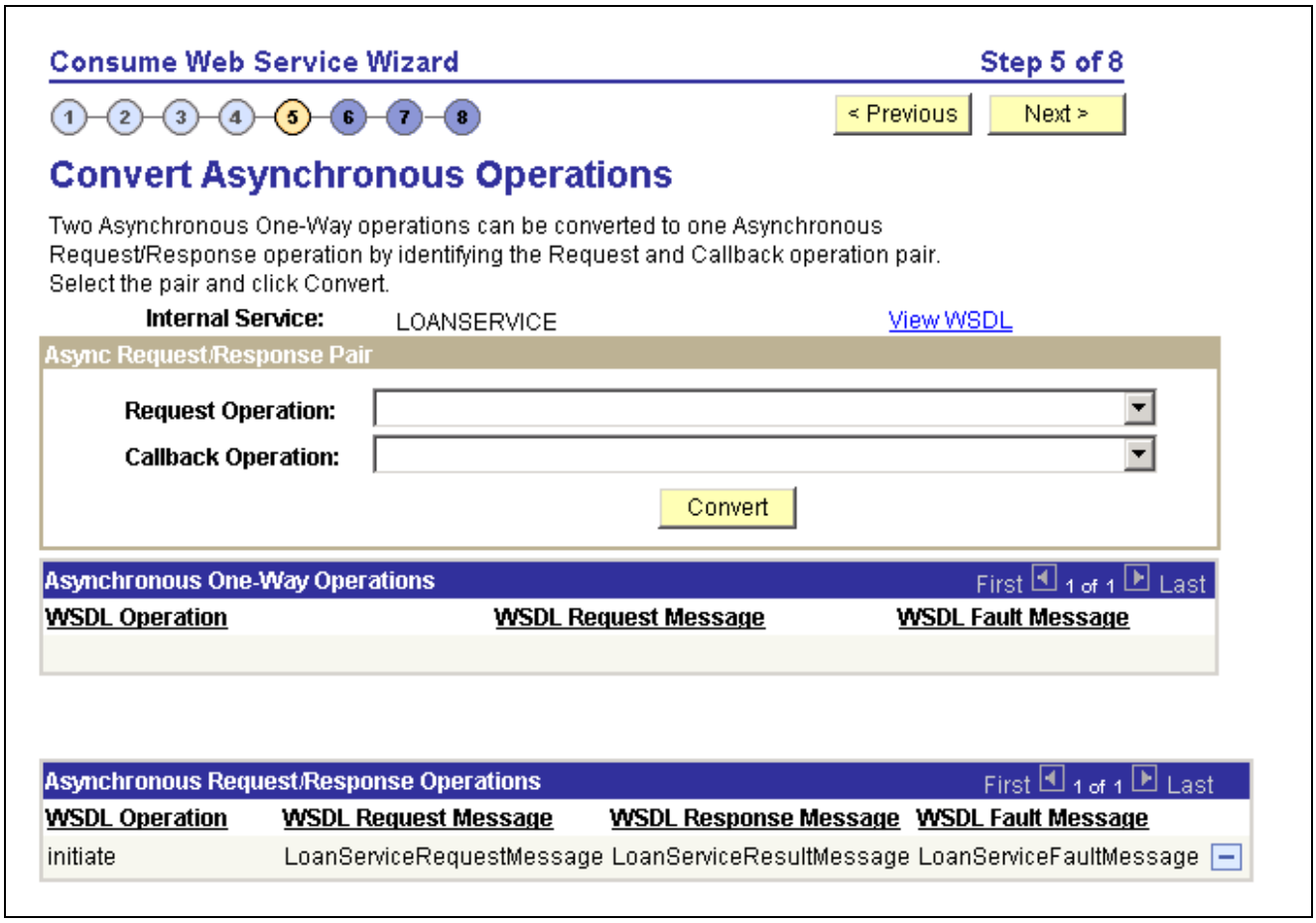
This page enables you to convert the two operations into a single asynchronous request/response operation type.

WSDL specification 1.1 has no standards for specifying an asynchronous request/response operation. Hence, the Consume Web Service wizard is not able to automatically detect such operations in a WSDL 1.1 document.

To make this conversion, you must specify the request operation and the callback operation, using the Request Operation and Callback Operation fields on the page. The system populates the possible values to select in each field from the operation selected.

After you make the conversion the new asynchronous request/response operation appears in the Asynchronous Request/Response Operations section of the page. The following example shows the Consume Web Service-Convert Asynchronous Operations page after making such a conversion:

Copyright © 1988-2007, Oracle. All rights reserved.



Consume Web Service-Convert Asynchronous Operations page after the operation conversion

To convert two one-way asynchronous operations into one asynchronous request/response operation:

1. From the Request Operation dropdown list box, select the request operation.
2. From the Callback Operation dropdown list box, select the callback operation.
3. Click the Convert button.

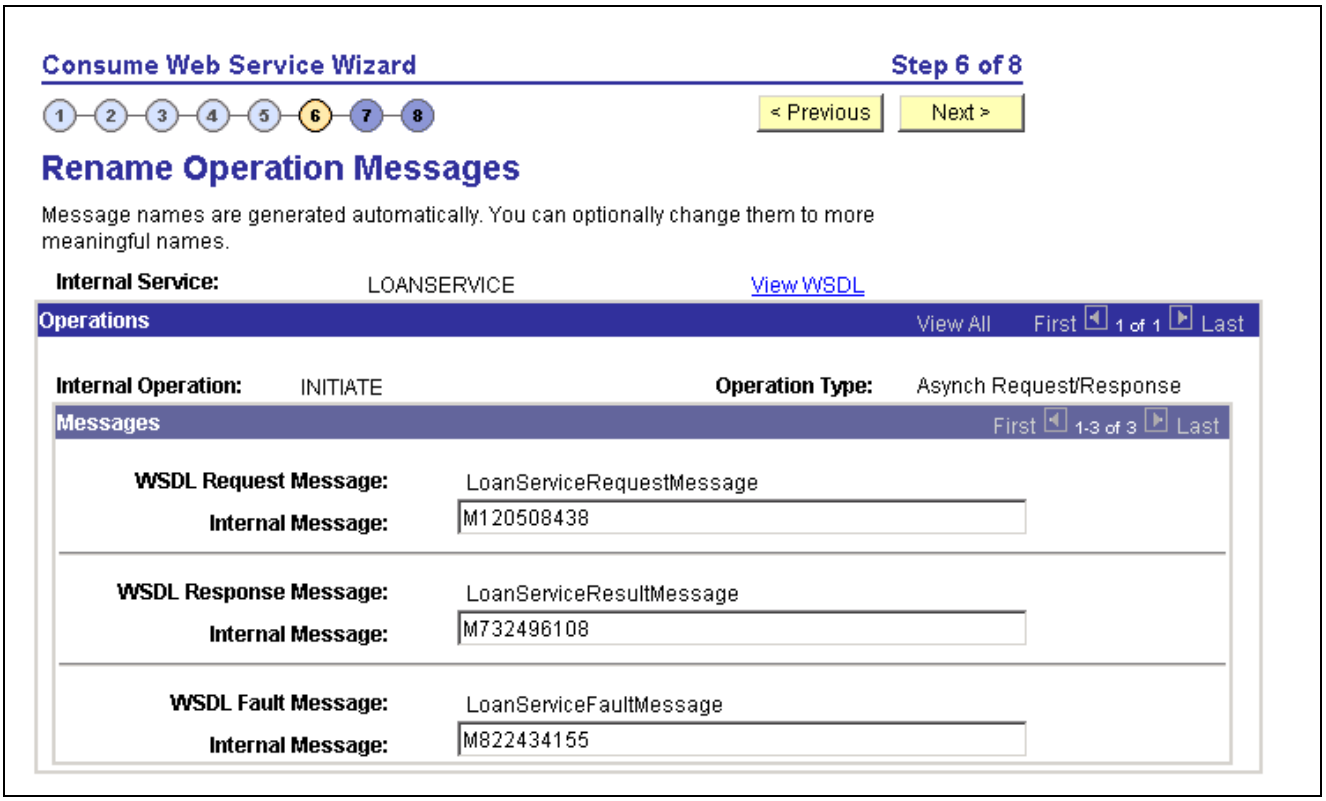
The single operation appears in the Asynchronous Request/Response Operations grid at the bottom of the page.

Clicking the minus button (-) at the end of a data row in the Asynchronous Request/Response grid undoes the conversion.

Step 6: Rename Operation Messages

The Consume Web Service-Rename Operation Messages page is shown in the following example:

Copyright © 1988-2007, Oracle. All rights reserved.

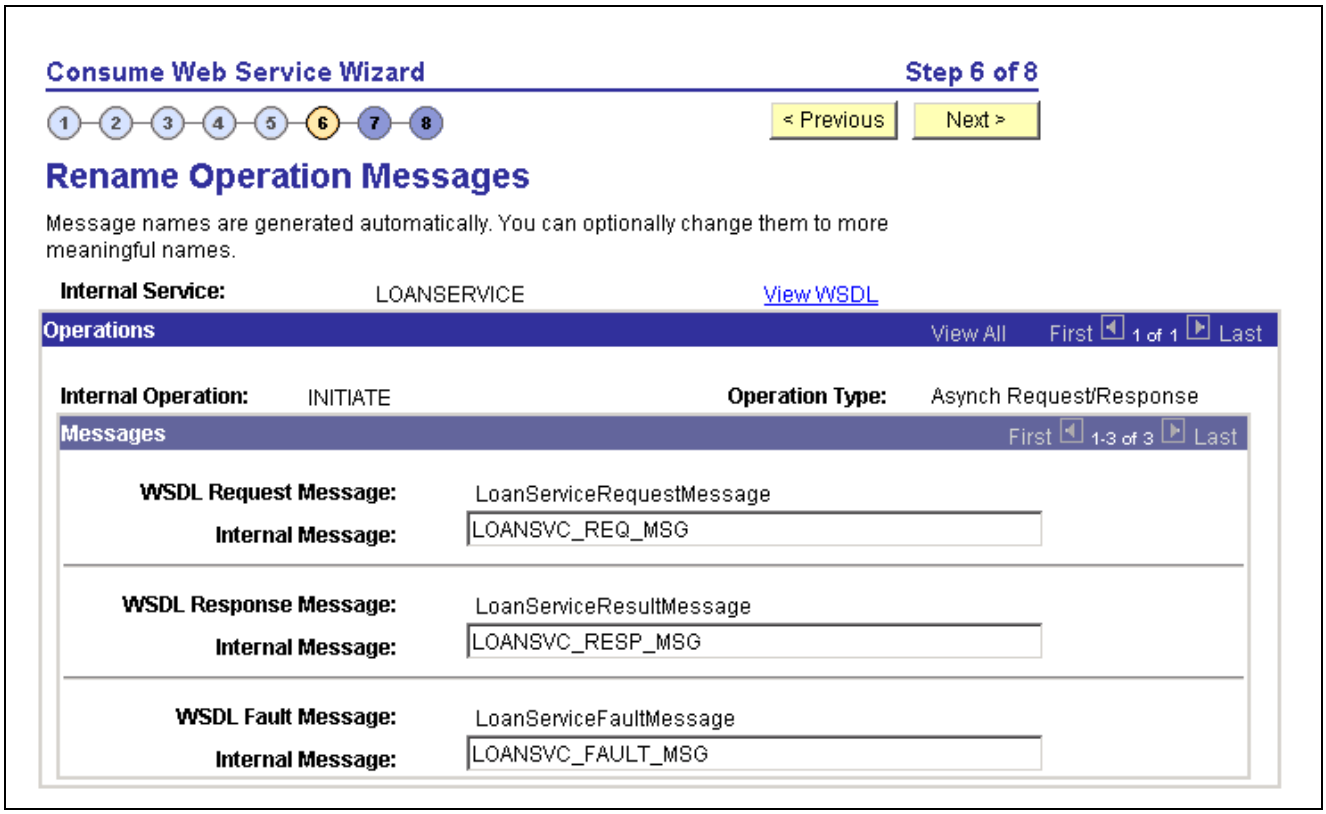


Consume Web Service-Rename Operation Messages page

When the system creates request, response and fault messages from the consumed service, it provides them with system-generated names. The previous example shows system-generated names appearing for all messages.

Use the page to rename the messages to more meaningful names. The following example shows all messages renamed:

Copyright © 1988-2007, Oracle. All rights reserved.



Renamed service operation messages

To rename operation messages:

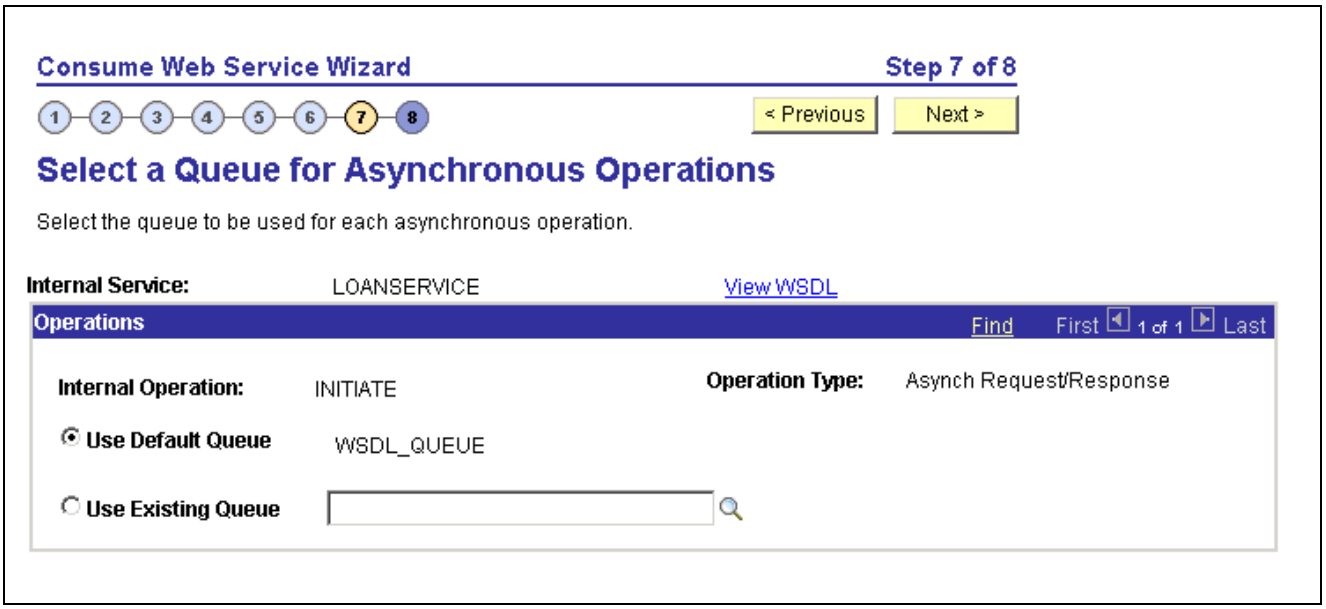
1. Clear the system-generated name from a message name field.
2. Enter a new name in the field.
3. Click the Next button to proceed to the next step in the wizard.

The system checks whether the user-entered message name already exists in the database and displays an error if the name exists.

Step 7: Select a Queue for Asynchronous Operations

The Select a Queue for Asynchronous Operations page appears only when you are consuming asynchronous one-way and asynchronous request/response operations. The following example shows the page:

Copyright © 1988-2007, Oracle. All rights reserved.



Consume Web Service-Select a Queue for Asynchronous Operations page

Note. This page appears only when asynchronous operations are being consumed from the WSDL document.

Use the Consume Web Service-Select a Queue for Asynchronous Operations page to select a service operation queue for an asynchronous service operation.

PeopleSoft delivers a queue, *WSDL_QUEUE*, to which it assigns asynchronous service operations by default. However, you can select a different queue to use.

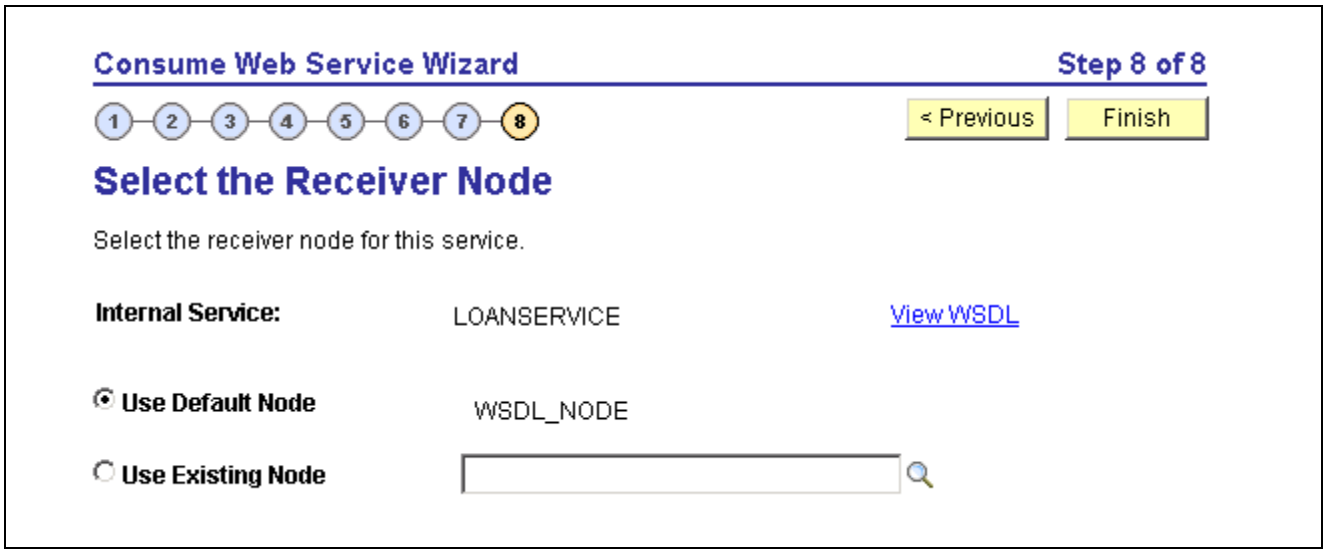
To select a queue for asynchronous operations:

1. Click the Use Existing Queue radio button.
2. From the Use Existing Queue dropdown list box, select the queue to use for the service operation.
3. Click the Next button to proceed to the next step in the wizard.

Step 8: Select the Receiver Node

Use the Consume Web Service-Select the Receiver Node page to select the receiving node for the service. The following example shows the page:

Copyright © 1988-2007, Oracle. All rights reserved.



Consume Web Service-Select the Receiver Node page

PeopleSoft delivers a node, *WSDL_NODE*, that the system uses as the receiving node by default. However, you can select a different receiving node.

If the you use the Default node as the receiver node, the system adds connector property overrides to the default node in the generated service operation routing

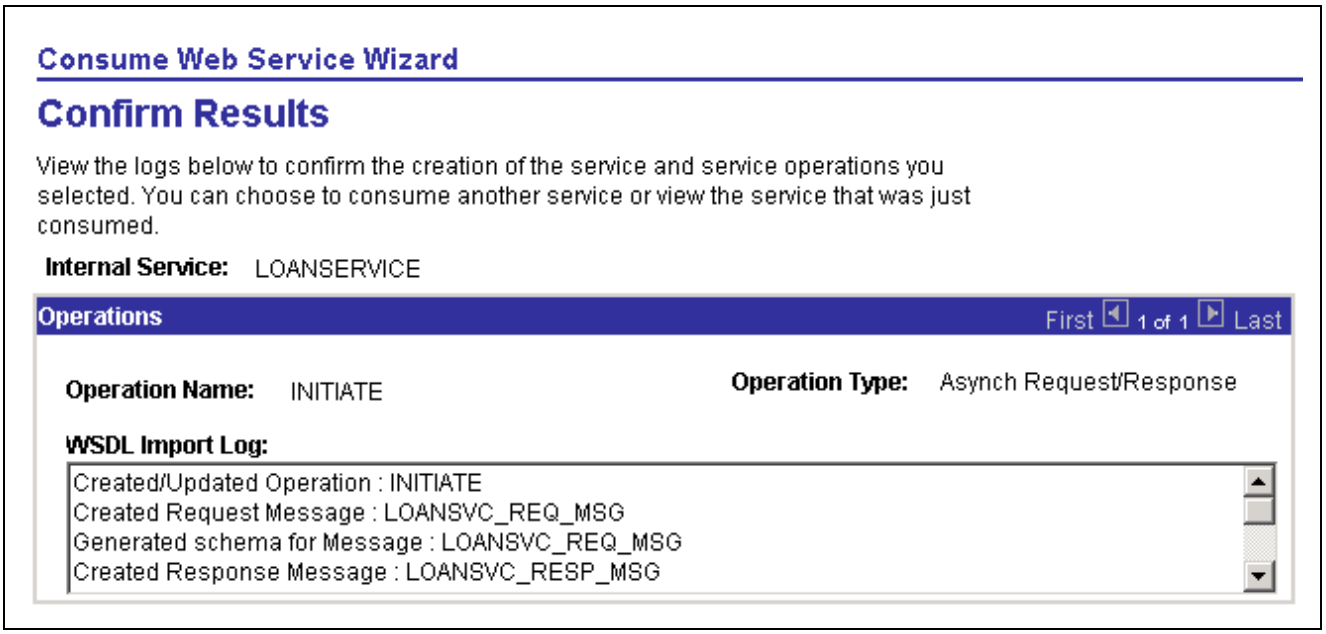
To select a receiving node for a service operation:

1. Click the Use Existing Node radio button.
2. From the Use Existing Node dropdown list box, select the receiving node to use for the service operation.
3. Click the Finish button to proceed to the next step in the wizard.

Confirm and View Results

The final page in the Consume Web Service wizard is the Consume Web Service-Confirm Results page shown in the following example:

Copyright © 1988-2007, Oracle. All rights reserved.



Consume Web Service-Confirm Results page

The Consume Web Service-Confirm Results page provides a WSDL Import Log. The WSDL Import log provides a summary of the WSDL import and lists the integration metadata created. The following example shows the contents of the WSDL Import Log for the example shown in this chapter of consuming a service:

```
Created/Updated Operation : INITIATE
Created Request Message : LOANSVC_REQ_MSG
Generated schema for Message : LOANSVC_REQ_MSG
Created Response Message : LOANSVC_RESP_MSG
Generated schema for Message : LOANSVC_RESP_MSG
Created Fault Message : LOANSVC_FAULT_MSG
Generated schema for Message  : LOANSVC_FAULT_MSG
Created Routing: ~IMPORTED~14857 for Operation: INITIATE
Created Operation Version: V1
```

The Consume Web Service-Confirm Results page also features the following page elements:

View Consumed Service Click the button to open the consumed service in the Services component.

See Chapter 14, "Managing Services," Accessing and Viewing Service Definitions, page 282.

Consume Another Service Click the button to go back to step 1 of the Consume Web Service wizard and consume another service.

Accessing Integration Metadata for Consumed Services

This section discusses how to access integration metadata for consumed services.

Copyright © 1988-2007, Oracle. All rights reserved.

Pages Used to Access Integration Metadata for Consumed Services

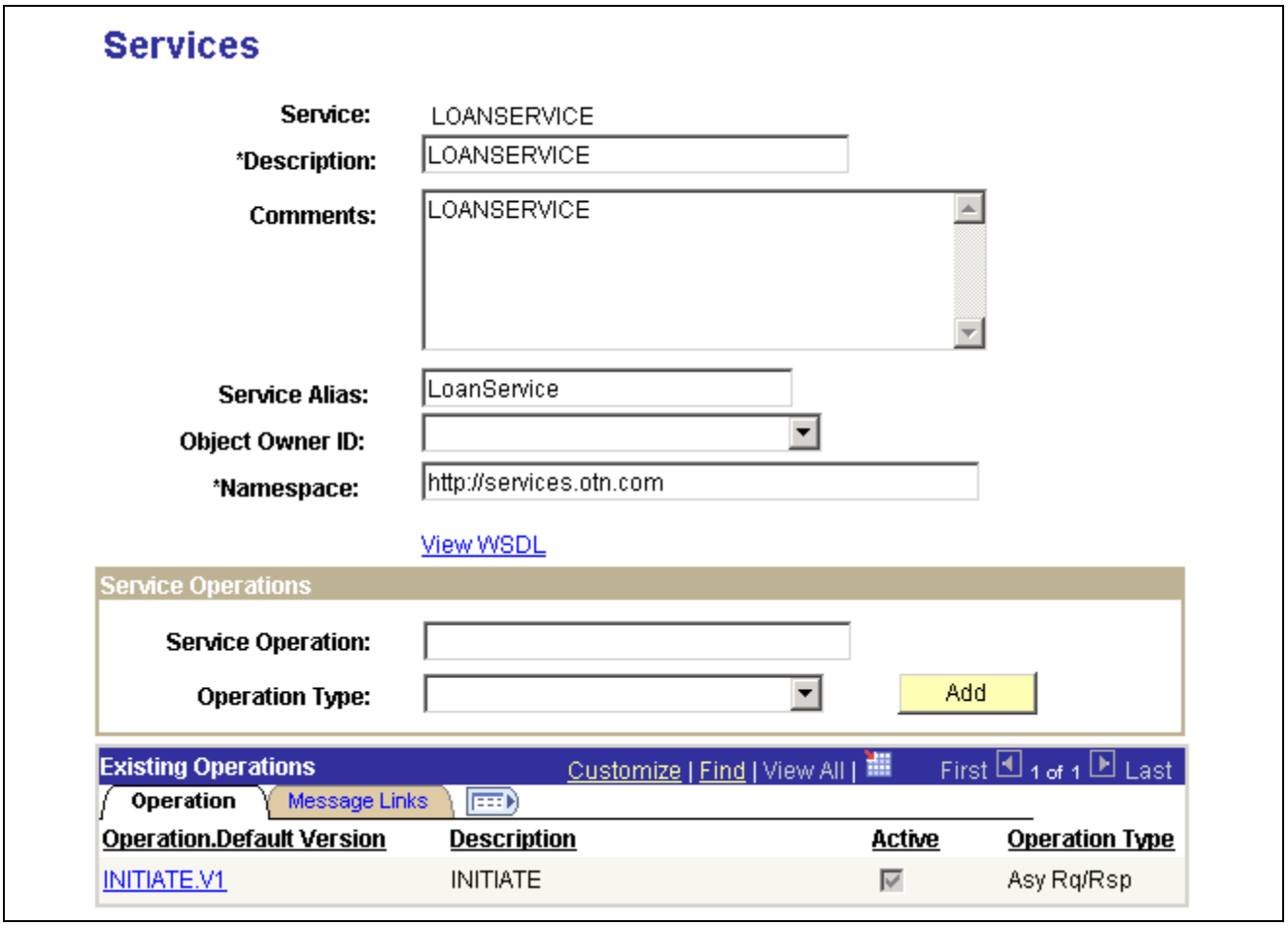
Page Name	Object Name	Navigation	Usage
Services page	IB_SERVICEDEFN	Use one of the following methods to access the Services page • On the Consume Web Service-Results page, click the View Consumed Service link • Select PeopleTools, Integration Broker, Integration Setup, Services.	Use the page to view service details, access service operations associated with the service and view WSDL documents generated for the service.

Accessing Integration Metadata for a Consumed Service

After using the Consume Web Service wizard to consume a service into the PeopleSoft system, use the Service component to access , view and modify the integration metadata created.

The following example shows the service definition for the *LOANSERVICE* service created with the Consume Web Service wizard.

Copyright © 1988-2007, Oracle. All rights reserved.



Service definition for the service LOANSERVICE

The example shows that when consuming a service, the PeopleSoft system creates active versioned service operations for all operations of the service. In addition, the system saves the consumed WSDL documents for the service operations and you can view the WSDL documents by clicking the View WSDL link.

In the Existing Operations section, click the name of the service operation to open the Service Operations component. Use the Service Operations component to view and modify service operation data and message data, add handlers, and view and modify routing definitions created by the system.

See Also

Chapter 14, “Managing Services,” Accessing and Viewing Service Definitions, page 282

Copyright © 1988-2007, Oracle. All rights reserved.

 Copyright © 1988-2007, Oracle. All rights reserved.

CHAPTER 25

Integrating with BPEL Process-Based Services

This chapter provides an overview of integrating with Business Process Execution Language (BPEL) processes, lists prerequisites for integrating with BPEL processes, and discusses how to:

- Configure the PeopleSoft-delivered BPEL node.
- Consume BPEL process services.
- Provide PeopleSoft services to BPEL processes.

Understanding Integrating with BPEL Processes

PeopleSoft enables you to integrate with BPEL processes.

Oracle BPEL Process Manager

You can use any BPEL runtime engine for integrations with BPEL processes. The developer version of Oracle BPEL Process Manager 10.1.2.0.2 for OC4J is the BPEL engine that is referenced in this chapter.

Oracle BPEL Process Manager 10.1.2.0.2 for OC4J is a standards-based process orchestration and integration solution that provides the foundation for developing and deploying a services-oriented and web service-enabled integrations. The Oracle BPEL Process Manager provides native support for BPEL 1.1 in combination with broad web service standard support to ensure interoperability with other BPEL and web service-compliant solutions.

Please note that PeopleSoft Integration Broker includes all necessary components for XML mapping/XSL transformation.

See Also

http://www.oracle.com/technology/index.html

Oracle JDeveloper 10g Release 2 (10.1.2) Installation Guide

PeopleTools Installation Guide for your database

Oracle BPEL Process Manager Quick Start Guide

Oracle BPEL Process Manager Developer's Guide

PeopleSoft-Delivered Application Classes for BPEL Integrations

PeopleSoft provides several application classes for launching and controlling BPEL process instances. The following application classes are located in the PT_BPEL application package and are accessible in PeopleSoft Application Designer:

Copyright © 1988-2007, Oracle. All rights reserved.

- AsyncFFSend

  This class provides the onSend handler implementation when calling a BPEL web service in an asynchronous fire and forget fashion.

- BPELUtil

  This class provides utility methods for interacting with BPEL processes from PeopleSoft systems.

- IBUtil

  This class provides utility methods to access PeopleSoft Integration Broker metadata.

See Also

Enterprise PeopleTools 8.49 PeopleBook: PeopleCode API Reference, "BPEL Classes"

Monitoring Integrations

As with all integrations, PeopleTools provides the following tools and logs for monitoring, tracing, and debugging integrations on PeopleSoft systems:

- Service Operations Monitor
- Integration gateway logs
- PeopleSoft application server logs

In addition, your BPEL runtime engine may provide additional tools for monitoring integrations. For example, Oracle BPEL Process Manager provides the Oracle BPEL Process Manager Console for managing, administering, and debugging processes that are deployed on the BPEL server. In addition, the Oracle Application Server (OAS) logs may also provide additional details. Check your BPEL runtime engine documentation for additional information about monitoring tools that are provided.

See Also

Chapter 21, "Using the Service Operations Monitor," page 413

Chapter 22, "Managing Error Handling, Logging, Tracing, and Debugging," page 497

Prerequisites for Integrating with BPEL Processes

For creating integrations with BPEL processes, you must have PeopleSoft Integration Broker configured and running.

Note. This section discusses prerequisite configuration steps on the PeopleSoft system for creating integrations with BPEL processes. Check your BPEL runtime engine software documentation for setup and configuration steps that you must perform on the runtime engine prior to performing integrations.

The following list is a partial checklist of items to configure:

- When configuring the integrationGateway.properties file, be sure to set the ig.isc.serverURL property equal to the name of the machine running the integration engine.
- When configuring the PeopleTools application server, set the PUB/SUB option to *Yes*. This value is required for asynchronous integrations.

 Copyright © 1988-2007, Oracle. All rights reserved.

- On the Integration Broker Quick Configuration page, be sure to activate the application server domain by setting the Domain Status to *Active*.
- On the Integration Broker Services Configuration page, be sure to set the service namespace, the schema namespace, and the target location.
- To load files into PeopleTools from the file system, set PS_FILEDIR and PS_SERVDIR in the system variables on your machine.
- PeopleSoft delivers a node, *BPEL*, specifically for BPEL 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. Steps to configure the *BPEL* node are provided elsewhere in this chapter.

The *Understanding Creating and Implementing Integrations* chapter of this PeopleBook contains additional information about creating and implementing integrations on PeopleSoft systems.

See Also

Chapter 4, "Understanding Creating and Implementing Integrations," page 31

Chapter 5, "Using the Integration Broker Quick Configuration Page," page 37

Configuring the PeopleSoft-Delivered BPEL Node

PeopleTools delivers a node called *BPEL* for use for BPEL integrations when you are using Oracle BPEL Process Manager as the runtime engine.

Note. You must configure the BPEL node only if you are using Oracle BPEL Process Manager as the BPEL runtime engine.

To configure BPEL node properties:

1. Select PeopleTools, Integration Broker, Integration Setup, Nodes.
2. Search for and open the node *BPEL*.

   The Node Definitions page appears.
3. Click the Properties link at the bottom of the page.

   The Node Properties page appears.

   Note. To view the complete BPEL property name in any of the Property Name fields, insert your cursor in a field and use your keyboard arrow keys to scroll through and view the complete field name.
4. Set the value of the BPELCONSOLEURL property to the URL of the BPEL console.
5. Set the BPELDOMAIN property to the name of the domain that you are using for the BPEL server.
6. Click the OK button.

Note. Do not set any values for the BPELDOMAINPWD property. This property is reserved for future use.

See Also

Chapter 19, "Adding and Configuring Nodes," page 361

Copyright © 1988-2007, Oracle. All rights reserved.

Consuming BPEL Process–Based Services

This section provides an overview of consuming BPEL process-based services and discusses how to:

- Deploy BPEL processes.
- Import WSDL documents from BPEL processes.
- Consume synchronous BPEL operations.
- Consume asynchronous request/response BPEL operations.
- Consume asynchronous one-way BPEL operations.

In addition, the end of this section provides development considerations for consuming BPEL web services.

Understanding Consuming BPEL Process-Based Services

This section provides overview information about consuming BPEL process-based services.

Development Considerations

When consuming BPEL process-based services, consider that :

- XML content representing the payload must be constructed carefully and exactly.
- XML namespace in the top-level element of the XML fragment representing the payload is obtained from the WSDL of the BPEL service operation.
- The LaunchSyncBPELProcess and LaunchASyncBPELProcess methods do not specify the actual endpoints to which a message is sent. This endpoint is inferred by PeopleSoft Integration Broker at runtime based on the active routing that is associated with the service operation. An exception is raised if no routing exists or if more than one active routing exists.
- When using the LaunchSyncBPELProcess and LaunchASyncBPELProcess methods, you must configure the routings for the message so that exactly one active routing exists for a message. Failure to supply exactly one routing results in a runtime exception.
- PeopleSoft Integration Broker automatically maps an external service operation to an internal PeopleSoft operation. If you attempt to import an external operation that contains the same name as an internal operation that already exists, PeopleSoft Integration Broker provides the new operation with a unique name and new metadata that maps the internal name to the external name. Make sure to use the unique internal name when calling any of the application class methods in the PT_BPEL application package.
- For the asynchronous request/response operations, you must select the correct callback (response) operation for a given service request. This is achieved during the Convert step in the Consume Web Service wizard.

Deploying BPEL Processes and Importing WSDL Documents

The first two steps for consuming any type of BPEL service are to deploy the BPEL process and import the generated WSDL document into the PeopleSoft system.

The following two sections discuss performing these tasks and must be completed before consuming BPEL operations.

 Copyright © 1988-2007, Oracle. All rights reserved.

Deploying BPEL Processes

The first step in consuming a BPEL process-based service is to deploy the BPEL process in the BPEL runtime engine, thereby generating a WSDL document. See the documentation for your BPEL runtime engine for information about performing this task.

The PeopleSoft system can consume WSDL documents from a UDDI repository, WSDL URL, WSIL URL, or file.

When deploying a BPEL process, note the URL or file location of the WSDL document. You must provide the document location when consuming the WSDL into the PeopleSoft system.

Consuming WSDL Documents from BPEL Processes

To consume WSDL from a BPEL process into the PeopleSoft system use the Consume Web Service component in the PeopleSoft Pure Internet Architecture.

Note. If Oracle BPEL Process Manager is your BPEL runtime engine, during the WSDL import process, select the *BPEL* node when prompted to select the receiving node.

Make a note of the web service operation for the process that you are importing as you will call this operation in PeopleCode.

See Also

Chapter 23, “Providing Services,” page 507

Consuming Synchronous BPEL Operations

This section provides an overview of consuming synchronous BPEL operations and discusses how to:

- Create synchronous BPEL requests.
- Invoke synchronous BPEL operations.
- Process synchronous BPEL responses.

Understanding Consuming Synchronous BPEL Operations

Before consuming synchronous BPEL operations, you must deploy a BPEL process and import the generated WSDL document into the PeopleSoft system.

See Chapter 25, “Integrating with BPEL Process-Based Services,” Deploying BPEL Processes, page 557 and Chapter 25, “Integrating with BPEL Process-Based Services,” Consuming WSDL Documents from BPEL Processes, page 557.

Creating Synchronous BPEL Requests

In this step, you use the PeopleSoft-delivered BPEL application classes to create a request message and initialize it with the message content, or payload. The payload is the SOAP request (envelope) that will be sent to the BPEL process as a service.

The first step is to create a BPELUtil object.

```
&bpel = create PT_BPEL:BPELUtil();
```

Copyright © 1988-2007, Oracle. All rights reserved.

Next, create the request message and specify the operation to invoke. In the following pseudo code example, *PROCESS* is the operation to be invoked.

```
&payload = "<?xml version='1.0'?>
 <SyncCalcProcessRequest xmlns='http://xmlns.oracle.com/SyncCalc'>
 <messageid>TestId::0123456789</messageid><op>+</op><input1>9</input1>
 <input2>3</input2></SyncCalcProcessRequest>";

&xml = CreateXmlDoc(&payload);
&msg = CreateMessage(Operation.PROCESS, %IntBroker_Request);
&msg.SetXmlDoc(&xml);
```

Invoking Synchronous BPEL Operations

To invoke a synchronous BPEL operation, use the LaunchSyncBPELProcess method of the BPELUtil application class.

The following example shows pseudo code for invoking a synchronous operation.

```
&reply = &bpel.LaunchSyncBPELProcess(&OP_NAME="PROCESS", &msg, "", "");
```

Processing Synchronous BPEL Responses

The following PeopleCode example shows sample pseudo code for processing a synchronous BPEL response.

```
If All(&reply) Then
   &responseStr = &reply.GenXMLString();
   End-If;
```

Example: Consuming Synchronous BPEL Operations

The following pseudo code provides an example of all the PeopleCode discussed earlier in this section.

```
import PT_BPEL:*;

Local PT_BPEL:BPELUtil &bpel;
Local string &url,  &transactionId,
&payload, &responseStr, &OP_NAME;
Local Message &msg, &reply;
Local XmlDoc &xml;

/* --- creating a BPELUtil object---*/
&bpel = create PT_BPEL:BPELUtil();

/* --- setting operation name --- */
&OP_NAME="PROCESS";
&transactionId = "TestId::0123456789";

&payload = "<?xml version='1.0'?><SyncCalcProcessRequest xmlns=
'http://xmlns.oracle.com/SyncCalc'><messageid>TestId::0123456789</messageid>
<op>+</op><input1>9</input1><input2>3</input2></SyncCalcProcessRequest>";
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
&xml = CreateXmlDoc(&payload);
&msg = CreateMessage(Operation.PROCESS, %IntBroker_Request);
&msg.SetXmlDoc(&xml);

&reply = &bpel.LaunchSyncBPELProcess(&OP_NAME, &msg, "", "");
If All(&reply) Then
   &responseStr = &reply.GenXMLString();
   End-If;
&url = &bpel.GetSyncBPELProcessInstanceUrl("BPEL", &transactionId);
```

See Also

Enterprise PeopleTools 8.49 PeopleBook: PeopleCode API Reference, "BPEL Classes"

Consuming Asynchronous Request/Response BPEL Operations

This section provides an overview of consuming asynchronous request/response BPEL operations and discusses how to:

- Create an asynchronous request/response BPEL request.
- Invoke an asynchronous request/response BPEL operation.
- Create a handler to process the asynchronous request/response BPEL response.
- Add a handler to a service operation to process the asynchronous request/response BPEL response.

This section also features a comprehensive example of the PeopleCode discussed in this section.

Understanding Consuming Asynchronous Request/Response BPEL Operations

Before consuming asynchronous request/response BPEL operations, you must deploy a BPEL process and import the generated WSDL document into the PeopleSoft system.

When you consume an Asynchronous Request/Response WSDL, you must identify the request and callback operations correctly in the Consume Web Service wizard.

See Chapter 25, "Integrating with BPEL Process-Based Services," Deploying BPEL Processes, page 557 and Chapter 25, "Integrating with BPEL Process-Based Services," Consuming WSDL Documents from BPEL Processes, page 557.

Creating Asynchronous Request/Response BPEL Requests

In this step, you use the PeopleSoft-delivered BPEL application classes to create a request message and initialize it with the message content, or payload. The payload is the SOAP request (envelope) that will be sent to the BPEL process as a service.

```
&bpel = create PT_BPEL:BPELUtil();
```

Next, create the request message and specify the operation to invoke. In the following pseudo code example, *CALCULATE* is the operation to be invoked:

```
&payload = "<?xml version='1.0'?>
<ASyncCalcProcessRequest  xmlns='http://xmlns.oracle.com/ASyncCalc'>
<op>+</op><input1>9</input1><input2>3</input2></ASyncCalcProcessRequest>";
```

Copyright © 1988-2007, Oracle. All rights reserved.

```
&xml = CreateXmlDoc(&payload);
&msg = CreateMessage(Operation.CALCULATE, %IntBroker_Request);
&msg.SetXmlDoc(&xml);
```

Invoking Asynchronous Request/Response BPEL Operations

To invoke an asynchronous BPEL operation, use the LaunchASyncBPELProcess method of the BPELUtil application class.

The following example shows pseudo code for invoking an asynchronous operation.

```
&responseStr = &bpel.LaunchASyncBPELProcess(&OP_NAME="CALCULATE", &msg, "", "");
```

Creating Handlers to Process Asynchronous Request/Response BPEL Responses

When you imported the asynchronous request/response BPEL WSDL document into the PeopleSoft system, the system automatically created a request message and response message for each service operation that is contained in the WSDL document.

To process the response, you must create a handler and add it to the service operation record.

To create a handler for the response to an asynchronous request/response operation, use PeopleSoft Application Designer to extend the PS_PT:Integration:INotificationHandler application class and use the OnResponse method to code the response. When PeopleSoft Integration Broker receives the response for the service operation, Integration Broker will use the handler code to process the response.

The following example shows a sample pseudo-code application class that is designed to handle the response:

```
import PS_PT:Integration:INotificationHandler;

class ASyncTestHandler implements PS_PT:Integration:INotificationHandler
   method ASyncTestHandler();
   method OnNotify(&MSG As Message);
end-class;

/* constructor */
method ASyncTestHandler
end-method;

method OnNotify
   /+ &MSG as Message +/
   /+ Extends/implements PS_PT:Integration:INotificationHandler.OnNotify +/

   Local File &MYFILE;
   &MYFILE = GetFile("C:\temp\item.txt", "W", %FilePath_Absolute);

   If &MYFILE.IsOpen Then
      &MYFILE.WriteLine("--- Response Received ---");
      &MYFILE.WriteLine(&MSG.GenXMLString());
      &MYFILE.Close();
   End-If;
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
end-method;
```

Adding Handlers to Operations to Process Asynchronous Request/Response BPEL Responses

After you create the response handler, you must add it to the service operation definition. To perform this task, use the Service Operations–Handlers page. To access this page, select PeopleTools, Integration Broker, Integration Setup, Service Operations, and click the Handlers tab.

Example: Consuming Asynchronous Request/Response BPEL Operations

The following pseudo code provides a full example of all the PeopleCode discussed earlier for creating the asynchronous request and invoking the service operation.

```
import PT_BPEL:*;

Local PT_BPEL:IBUtil &ibutil;
Local PT_BPEL:BPELUtil &bpel;
Local string &url, &domain, &pwd, &opType, &asyncUrl, &transactionId,
&payload, &responseStr, &OP_NAME;
Local number &routings;
Local Message &msg, &reply;
Local XmlDoc &xml;

/* --- creating a BPELUtil object --- */
&bpel = create PT_BPEL:BPELUtil();

/* --- setting operation name --- */
&OP_NAME = "CALCULATE";

&payload = "<?xml version='1.0'?><ASyncCalcProcessRequest xmlns=
'http://xmlns.oracle.com/ASyncCalc'><op>+</op><input1>9</input1>
<input2>3</input2></ASyncCalcProcessRequest>";

&xml = CreateXmlDoc(&payload);
&msg = CreateMessage(Operation.CALCULATE, %IntBroker_Request);
&msg.SetXmlDoc(&xml);

&responseStr = &bpel.LaunchASyncBPELProcess(&OP_NAME, &msg, "", "");
&url = &bpel.GetASyncBPELProcessInstanceUrl(&responseStr);
```

See Also

Chapter 15, "Managing Service Operations," Adding Handlers to Service Operations, page 301

Enterprise PeopleTools 8.49 PeopleBook: PeopleCode API Reference, "BPEL Classes"

Copyright © 1988-2007, Oracle. All rights reserved.

Consuming Asynchronous Fire-and-Forget (One-Way) BPEL Operations

This section provides an overview of consuming asynchronous fire-and-forget BPEL operations and discusses how to:

- Create an asynchronous fire-and-forget BPEL requests.
- Invoke an asynchronous fire-and-forget BPEL operation.
- Add handlers to assign correlation IDs to requests.

Note. BPEL asynchronous fire-and-forget operations correspond to asynchronous one-way operations in PeopleSoft systems.

This section also features a comprehensive example of the PeopleCode discussed in this section.

Understanding Consuming Asynchronous Fire-and-Forget BPEL Operations

Before consuming asynchronous fire-and-forget BPEL operations, you must deploy a BPEL process and import the generated WSDL document into the PeopleSoft system.

See Chapter 25, "Integrating with BPEL Process-Based Services," Deploying BPEL Processes, page 557 and Chapter 25, "Integrating with BPEL Process-Based Services," Consuming WSDL Documents from BPEL Processes, page 557.

Creating Asynchronous Fire–and–Forget BPEL Requests

In this step, you use the PeopleSoft-delivered BPEL application classes to create a request message and initialize it with the message content, or payload. The payload is the SOAP request (envelope) that will be sent to the BPEL process as a service.

The first step is to create a BPELUtil object.

```
&bpel = create PT_BPEL:BPELUtil();
```

Next, create the request message and specify the operation to invoke. In the following pseudo code example, *FIREANDFORGET* is the operation to be invoked:

```
&payload = "<?xml version='1.0'?>
<ASyncFFProcessRequest xmlns='http://xmlns.oracle.com/ASyncFF'>
<input>test</input></ASyncFFProcessRequest>";

&xml = CreateXmlDoc(&payload);
&msg = CreateMessage(Operation. FIREANDFORGET, %IntBroker_Request);
&msg.SetXmlDoc(&xml);
```

Invoking Asynchronous Request/Response BPEL Operations

To invoke an asynchronous BPEL operation, use the LaunchASyncBPELProcess method of the BPELUtil application class.

The following example shows pseudo code for invoking a synchronous operation:

```
&responseStr = &bpel.LaunchASyncBPELProcess(&OP_NAME, &msg, "", "");
```

 Copyright © 1988-2007, Oracle. All rights reserved.

Adding Handlers to Assign Correlation IDs to Requests

You can create a handler to add a WSA_MessageID to the SOAP request header before the request is dispatched for the BPEL process.

To create a such a handler use PeopleSoft Application Designer to extend the PS_PT:Integration:ISend application class and use the AsyncFFSend method.

The following example shows a sample pseudo-code application class to perform this task:

```
import PS_PT:Integration:ISend;
import PT_BPEL:IBUtil;

class AsyncFFSend implements PS_PT:Integration:ISend
   method AsyncFFSend();
   method OnRequestSend(&MSG As Message) Returns Message;
end-class;

/* constructor */
method AsyncFFSend
end-method;

method OnRequestSend
   /+ &MSG as Message +/
   /+ Returns Message +/
   /+ Extends/implements PS_PT:Integration:ISend.OnRequestSend +/

   &MSG.IBInfo.WSA_MessageID = &MSG.TransactionId;
   Return &MSG;
end-method;
```

Example: Consuming Asynchronous Fire-and-Forget BPEL Processes

The following pseudo code provides a full example of all the PeopleCode discussed earlier for creating the asynchronous request and invoking the service operation.

```
import PT_BPEL:*;

Local PT_BPEL:BPELUtil &bpel;
Local string &url, &payload, &responseStr, &OP_NAME;
Local Message &msg, &reply;
Local XmlDoc &xml;

/* --- creating a BPELUtil object --- */
&bpel = create PT_BPEL:BPELUtil();

/* --- setting operation name --- */
&OP_NAME = "FIREANDFORGET";

&payload = "<?xml version='1.0'?><ASyncFFProcessRequest xmlns='http:
//xmlns.oracle.com/ASyncFF'><input>test</input></ASyncFFProcessRequest>";
```

Copyright © 1988-2007, Oracle. All rights reserved.

```
&xml = CreateXmlDoc(&payload);
&msg = CreateMessage(Operation.FIREANDFORGET, %IntBroker_Request);
&msg.SetXmlDoc(&xml);

&responseStr = &bpel.LaunchASyncBPELProcess(&OP_NAME, &msg, "", "");
&url = &bpel.GetASyncBPELProcessInstanceUrl(&responseStr);
```

See Also

Chapter 15, “Managing Service Operations,” Adding Handlers to Service Operations, page 301

Enterprise PeopleTools 8.49 PeopleBook: PeopleCode API Reference, “BPEL Classes”

Providing PeopleSoft Services to BPEL Processes

This section provides an overview of providing services to BPEL processes and discusses how to:

- Provide PeopleSoft synchronous operations to BPEL processes.
- Provide PeopleSoft asynchronous request/response operations to BPEL processes.

Understanding Providing PeopleSoft Services to BPEL Processes

This section provides overview information for providing services to BPEL processes.

Container Messages and Message Parts

If using container messages and message parts, create rowset-based parts in nonrowset-based containers. Doing so ensures that the XSchema that is generated for messages will not include PeopleTools-specific auditing information (PSCAMA). Typically, you do not want this auditing information to be included in the generated XSchema when integrating with BPEL processes

Provide Web Service Wizard

When providing services to systems using BPEL, consider the following guidelines:

- The providing web services step using the Provide Web Service wizard described in this section is for synchronous and asynchronous request/response operations and is required only for an external client to be able to consume a PeopleSoft service.
- When providing the service, the WSDL document is exported to the WSDL repository in the PeopleSoft Pure Internet Architecture. Optionally, you can select to export the WSDL to a UDDI repository. PeopleSoft uses relative path URL for schemas referenced in the WSDL documents the system generates.
- After generating the WSDL document, carefully inspect the results using the WSDL Generation Log (the last page of the Provide Web Services wizard).

  The WSDL Generation Log contains the *Generated WSDL URL*. Copy this URL and store it somewhere carefully. You will need this WSDL URL later when calling the PeopleSoft-provided web service operation from the BPEL process.

 Copyright © 1988-2007, Oracle. All rights reserved.

See Also

Chapter 10, “Managing Messages,” Managing Container Messages, page 189

Chapter 23, “Providing Services,” page 507

Providing Synchronous PeopleSoft Operations to BPEL Processes

This section discusses how to:

- Build synchronous PeopleSoft services.
- Provide synchronous PeopleSoft services as WSDL documents.
- Invoke and start the service from the BPEL process.

Building Synchronous PeopleSoft Services

This section lists the steps for building synchronous PeopleSoft services. Detailed documentation for each step is provided elsewhere in this PeopleBook. See the end of this section for links to the appropriate documentation.

1. Define request and response messages to be associated with the service operation.

   Use the appropriate XSchema for each message.

   If using rowset-based messages you do not need to generate message schemas. If you are using nonrowset-based message you must import a schema so that there is a shape to which to code and so the BPEL process knows what the response should look like.

   Examples of XSchema for request and response messages are provided at the end of this section.

2. Create a handler to process the request message.

   Extend the PS_PT:Integration:IRequestHandler application class using the OnRequest method. The output of the handler will be communicated back as the response to the received request.

   An code example of an OnRequest handler is provided at the end of this section.

3. Create a new service.
4. Add a synchronous operation to the service.
   - Generate an any-to-local routing definition for the operation.
   - Add permissions to the service operation.
   - Add the OnRequest handler that you created in step 2 to the operation by referencing the package name, path, and class ID of the handler that you created in step 2.

See Chapter 12, “Sending and Receiving Messages,” page 205; Chapter 12, “Sending and Receiving Messages,” Messaging Handlers, page 218; Chapter 14, “Managing Services,” page 275 and Chapter 15, “Managing Service Operations,” Configuring Service Operation Definitions, page 298.

Providing Synchronous PeopleSoft Services as WSDL Documents

Use the Provide Web Services wizard to export the service and generate a WSDL document.

See Chapter 23, “Providing Services,” page 507.

Invoke and Start the Service from the BPEL Process

The last step in the process is to invoke and start the PeopleSoft-provided service from the BPEL process.

Copyright © 1988-2007, Oracle. All rights reserved.

See the documentation that is provided with your BPEL runtime engine for the steps that are necessary to accomplish this step.

Example 1: Providing Synchronous Operations – Sample Request Message

The following pseudo code shows an example of a request message:

```
<?xml version="1.0"?>
<xsd:schema targetNamespace="http://xmlns.oracle.com/Enterprise/Tools
/schemas/PSFTCALCREQUESTMESSAGE.V1" xmlns="http://xmlns.oracle.com
/Enterprise/Tools/schemas/PSFTCALCREQUESTMESSAGE.V1" xmlns:xsd="http:
//www.w3.org/2001/XMLSchema">
  <xsd:element name="PSFTCalcRequestMessage" type=
"PSFTCalcRequestMessage_Type⇒
Shape"/>
  <xsd:complexType name="PSFTCalcRequestMessage_TypeShape">
    <xsd:sequence>
      <xsd:element name="op" type="xsd:string"/>
      <xsd:element name="input1" type="xsd:decimal"/>
      <xsd:element name="input2" type="xsd:decimal"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

Example 2: Providing Synchronous Operations – Sample Response Message

The following pseudo code shows an example of a response message:

```
<?xml version="1.0"?>
<xsd:schema targetNamespace="http://xmlns.oracle.com/Enterprise/Tools
/schemas/PSFTCALCRESPONSEMESSAGE.V1" xmlns="http://xmlns.oracle.com
/Enterprise/Tools/schemas/PSFTCALCRESPONSEMESSAGE.V1" xmlns:xsd="http:
//www.w3.org/2001/XMLSchema">
  <xsd:element name="PSFTCalcResponseMessage" type=
"PSFTCalcResponseMessage_TypeShape"/>
  <xsd:complexType name="PSFTCalcResponseMessage_TypeShape">
    <xsd:sequence>
      <xsd:element name="result" type="xsd:decimal"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

Example 3: Providing Synchronous Operations – OnRequest Handler

The following pseudo code shows an example of creating an OnRequest handler by extending the PS_PT:Integration:IBRequestHandler application class:

```
import PS_PT:Integration:IRequestHandler;

class InboundSyncRequestHandler implements PS_PT:Integration:IRequestHandler
   method InboundSyncRequestHandler();
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
   method onRequest(&MSG As Message) Returns Message;
   method onError(&MSG As Message) Returns string;
end-class;

method InboundSyncRequestHandler
end-method;

method onRequest
   /+ &MSG as Message +/
   /+ Returns Message +/
   /+ Extends/implements PS_PT:Integration:IRequestHandler.OnRequest +/

   Local Message &response;
   Local File &MYFILE;
   Local XmlDoc &xml, &inxml;
   Local string &payload, &oper, &input1, &input2;
   Local array of XmlNode &nodes;
   Local XmlNode &node;

   &nodes = CreateArray(&node);
   &inxml = &MSG.GetXmlDoc();
   &nodes = &inxml.GetElementsByTagName("op");
   &oper = &nodes [1].NodeValue;
   &nodes = &inxml.GetElementsByTagName("input1");
   &input1 = &nodes [1].NodeValue;
   &nodes = &inxml.GetElementsByTagName("input2");
   &input2 = &nodes [1].NodeValue;

   &payload = "<?xml version='1.0'?><PSFTCalcResponseMessage xmlns=
'http://xmlns.oracle.com/Enterprise/Tools/schemas
/PSFTCALCRESPONSEMESSAGE.V1'><result>9</result></PSFTCalcResponseMessage>";

   &xml = CreateXmlDoc(&payload);
   &response = CreateMessage(Operation.PSFTCALCULATE, %IntBroker_Response);
   &response.SetXmlDoc(&xml);

   Return &response;
end-method;

method onError
   /+ &MSG as Message +/
   /+ Returns String +/
   /+ Extends/implements PS_PT:Integration:IRequestHandler.OnError +/

   Local integer &nMsgNumber, &nMsgSetNumber;
   Local string &str;

   &nMsgNumber = &MSG.IBException.MessageNumber;
   &nMsgSetNumber = &MSG.IBException.MessageSetNumber;
   &str = &MSG.IBException.DefaultText;
```

Copyright © 1988-2007, Oracle. All rights reserved.

```
   Return &str;
end-method;
```

Providing Asynchronous PeopleSoft Request/Response Operations to BPEL Processes

This section discusses how to:

- Build asynchronous request/response PeopleSoft services.
- Provide an asynchronous request/response PeopleSoft service as a WSDL document.
- Invoke and start the service from the BPEL process.

Building Asynchronous Request/Response PeopleSoft Services

This section lists the steps for building asynchronous request/response PeopleSoft services. Detailed documentation for each step is provided elsewhere in this PeopleBook. See the end of this section for links to the appropriate documentation.

1. Define request and response messages to be associated with the service operation.

   Use the appropriate XSchema for each message.

   Examples of XSchema for request and response messages are provided at the end of this section.

2. Create a handler to process the request message.

   Extend the PS_PT:Integration:INotificationHandler application class using the OnNotify method. The output of the handler will be communicated back as the response to the received request.

   A code example of an OnNotify handler is provided at the end of this section.

3. Create a new service.
4. Add an Asynch Request/Response operation to the service.
   - Generate an any-to-local routing definition for the operation.
   - Add permissions to the service operation.
   - Add the OnNotify handler that you created in step 2 to the operation by referencing the package name, path, and class ID of the handler.

See Chapter 12, “Sending and Receiving Messages,” page 205; Chapter 12, “Sending and Receiving Messages,” Messaging Handlers, page 218; Chapter 14, “Managing Services,” page 275 and Chapter 15, “Managing Service Operations,” Configuring Service Operation Definitions, page 298.

Providing Asynchronous PeopleSoft Services as WSDL Documents

Use the Provide Web Services wizard to export the service and generate a WSDL document.

See Chapter 23, “Providing Services,” page 507.

Invoke and Start the Service from the BPEL Process

The last step in the process is to invoke and start the PeopleSoft-provided service from the BPEL process.

See the documentation that is provided with your BPEL runtime engine for the steps that are necessary to accomplish this step.

 Copyright © 1988-2007, Oracle. All rights reserved.

Example 1: Providing Asynchronous Request/Response Operations – Sample Request Message

The following pseudo code shows an example of a request message:

```
<?xml version="1.0"?>
<xsd:schema targetNamespace="http://xmlns.oracle.com/Enterprise/Tools
/schemas/PSFTCALCREQUESTMESSAGE.V1" xmlns="http://xmlns.oracle.com
/Enterprise/Tools/schemas/PSFTCALCREQUESTMESSAGE.V1" xmlns:xsd="http:
//www.w3.org/2001/XMLSchema">
  <xsd:element name="PSFTCalcRequestMessage" type=
"PSFTCalcRequestMessage_TypeShape"/>
  <xsd:complexType name="PSFTCalcRequestMessage_TypeShape">
    <xsd:sequence>
      <xsd:element name="op" type="xsd:string"/>
      <xsd:element name="input1" type="xsd:decimal"/>
      <xsd:element name="input2" type="xsd:decimal"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

Example 2: Providing Asynchronous Request/Response Operations – Sample Response Message

The following pseudo code shows an example of a response message:

```
<?xml version="1.0"?>
<xsd:schema targetNamespace="http://xmlns.oracle.com/Enterprise/Tools
/schemas/PSFTCALCRESPONSEMESSAGE.V1" xmlns="http://xmlns.oracle.com
/Enterprise/Tools/schemas/PSFTCALCRESPONSEMESSAGE.V1" xmlns:xsd="http:
//www.w3.org/2001/XMLSchema">
  <xsd:element name="PSFTCalcResponseMessage" type=
"PSFTCalcResponseMessage_TypeShape"/>
  <xsd:complexType name="PSFTCalcResponseMessage_TypeShape">
    <xsd:sequence>
      <xsd:element name="result" type="xsd:decimal"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

Example 3: Providing Asynchronous Request/Response Operations – Sample OnNotify Handler

The following pseudo code shows an example of creating an OnNotify handler by extending the PS_PT:Integration:INotification application class:

```
import PS_PT:Integration:INotificationHandler;

class InboundASyncResponseHandler implements PS_PT:Integration:
INotificationHandler
   method OnNotify(&MSG As Message);
```

Copyright © 1988-2007, Oracle. All rights reserved.

```
end-class;

method OnNotify
   /+ &MSG as Message +/
   /+ Extends/implements PS_PT:Integration:INotificationHandler.OnNotify +/

   Local Message &response;
   Local string &payload;
   Local XmlDoc &xml;
   Local File &MYFILE;

&payload = "<?xml version='1.0'?><PSFTCalcResponseMessage xmlns=
'http://xmlns.oracle.com/Enterprise/Tools/schemas
/PSFTCALCRESPONSEMESSAGE.V1'><result>9</result></PSFTCalcResponseMessage>";

   &xml = CreateXmlDoc(&payload);
   &response = CreateMessage(Operation.PSFTASYNCCALCULATE,
%IntBroker_Response);
   &response.SetXmlDoc(&xml);
   &response.IBInfo.WSA_MessageID = &MSG.IBInfo.WSA_MessageID;
   &response.IBInfo.WSA_ReplyTo = &MSG.IBInfo.WSA_ReplyTo;

   %IntBroker.Publish(&response);
end-method;
```

 Copyright © 1988-2007, Oracle. All rights reserved.

CHAPTER 26

Integrating with ERP Systems

This chapter provides overviews of integrations with Enterprise Resource Planning (ERP) systems and iWay SOAPswitch and discusses how to:

- Start and stop iWay SOAPswitch.
- Log in to iWay SOAPswitch.
- Generate WSDL documents using the ERP connectors.

Understanding Integrations with ERP Systems

PeopleSoft enables you to create inbound and outbound integrations with ERP systems. You accomplish these integrations using a combination of PeopleSoft Integration Broker and the third-party tool, iWay SOAPswitch.

iWay SOAPswitch provides ERP adapters, or connectors, that enable you to generate WSDL from SAP, Oracle, and Siebel systems. You then use PeopleSoft Integration Broker to import the WSDL into PeopleSoft, create integration metadata, extend the appropriate application classes, and perform the integrations.

Understanding iWay SOAPswitch

iWay SOAPswitch is a wizard-driven product that enables you to expose software functionality using web services, enabling you to make web services available to major development environments, such as SAP, Oracle, and Siebel. SOAPswitch generates WSDL for web services, allowing for simplified client development. SOAPswitch accepts SOAP requests for web services, translates them into calls to the back-end system, and formulates SOAP replies based on Web back-end system responses.

This section discusses:

- Installing iWay SOAPswitch.
- Components.
- Delivered adapters.
- Operation types.
- Web services and notifications.
- Database installation.
- Jetty Servlet server.
- iWay SOAPswitch and adapter documentation.

Copyright © 1988-2007, Oracle. All rights reserved.

Installing iWay SOAPswitch

iWay SOAPswitch version 5.5.3.5 is an installation option that you can use during the PeopleSoft Pure Internet Architecture installation process.

Note. Check the *Enterprise PeopleTools 8.49 Installation Guide for your database platform* to verify that iWay SOAPswitch is supported for your environment.

During the installation process, you are prompted to install and specify a location for ERP connectors. Choosing to install the ERP connectors is the same as installing iWay SOAPswitch.

Components

iWay SOAPswitch comprises of three main components:

SOAPswitch Server	The SOAPswitch server listens for SOAP requests from web service consumers and forwards requests to back-end servers through SOAPswitch adapters.
Administration Console	The Administration Console enables you to configure SOAPswitch to expose new web services, manage security, and monitor logging activity.
Web Services Viewer	The Web Services Viewer enables you to explore published web services and provides a directory of exposed web services, in WSDL format, for use by service consumers and clients.

Delivered Adapters

The following adapters are provided with the iWay SOAPswitch product that is delivered with PeopleTools 8.49:

Note. iWay SOAPswitch uses the term *adapter* to refer to connector. The terms *SOAPswitch adapters* and *ERP connectors* are used interchangeably in this chapter.

J2EE Adapter	The iWay J2EE adapter enables you to access Enterprise Java Beans (EJBs) and Java classes for SOAPswitch web services. This adapter is used primarily for testing purposes.
Oracle Applications Database (OAP) Adapter	The iWay SOAPswitch OracleApps Adapter enables you to expose any stored procedure or customer table that runs on Oracle 10g as a web service. In addition, it enables you to access stored procedures, tables, and views for Oracle applications using interface tables. SOAPswitch focuses on Oracle Financials and Oracle Project as reference implementations for the two primary Oracle application methodologies. Use this adapter for sending and receiving service operations between PeopleSoft and Oracle systems.
SAP R/3 Adapter	The iWay SOAPswitch SAP R/3 Adapter enables you to access R/3 Function modules and ALE IDOCs. Use this adapter for sending and receiving service operations between PeopleSoft and SAP systems.

 Copyright © 1988-2007, Oracle. All rights reserved.

Siebel Adapter The iWay SOAPswitch Siebel Adapter enables you to access the Siebel eBusiness 2000 Enterprise Edition application components for use with SOAPswitch.

Use this adapter to send service operations from PeopleSoft systems to Siebel systems. Use the XML adapter to send service operations from Siebel systems to PeopleSoft systems.

XML Adapter The iWay SOAPswitch XML Adapter enables you to set up and access XML objects for use with SOAPswitch.

Operation Types

The following table maps PeopleSoft service operation types and routings to their equivalent iWay SOAPswitch operation types.

PeopleSoft Service Operation Type and Routing	Equivalent iWay SOAPswitch Operation Type
Operation type is asynchronous – one way. Routing definition specifies that the sender is local and that the receiver is iWay SOAPswitch.	Notifications operation.
Operation type is asynchronous – one way. Routing definition specifies that the sender is iWay SOAPswitch and the receiver is local.	One-way operation.
Operation type is synchronous. Routing definition specifies that the sender is local and that the receiver is iWay SOAPswitch.	Solicit-Response operation.
Operation type is synchronous. Routing definition specifies that the sender is iWay SOAPswitch and the receiver is local.	Request-Response operation.

The iWay SOAPswitch documentation provides more information about operation types.

See *iWay SOAPswitch Administration Guide, "Chapter 2: iWay SOAPswitch Overview," Operational Types in WSDLs.*

Web Services and Notifications

A web service that is defined on iWay SOAPswitch specifies inbound asynchronous or synchronous transactions to SOAPswitch.

Notifications specify outbound asynchronous or synchronous transactions from iWay SOAPswitch.

iWay SOAPswitch and Adapter Documentation

This PeopleBook provides summary information about tasks that you perform in iWay SOAPswitch and with the iWay SOAPswitch adapters, and provides cross-references to iWay SOAPswitch documentation whenever possible.

Copyright © 1988-2007, Oracle. All rights reserved.

To take full advantage of iWay SOAPswitch, you should thoroughly review the iWay SOAPswitch documentation.

Locating Documentation

You can access iWay SOAPswitch documentation from the SOAPswitch Administration Console.

1. Select PeopleTools, Integration Broker, Web Services, ERP Connectors.

   The iWay SOAPswitch Administration Console appears.
2. From the left navigation area, select Documentation.

Documentation is also available in PDF format in <ERPConnector_Install_Dir>\webapps\ssw\docs.

Documentation List

The following iWay SOAPswitch documentation is provided in HTML and PDF formats:

- *iWay SOAPswitch Getting Started Guide*
- *iWay SOAPswitch Administration Guide*
- *iWay SOAPswitch Managing Existing Web Services Guide*

The following iWay SOAPswitch adapter documentation is available.

- *iWay SOAPswitch J2EE™ Adapter Guide*
- *iWay SOAPswitch Oracle™ Applications Adapter Guide*
- *iWay SOAPswitch SAP™ R/3™ Adapter Guide*
- *iWay SOAPswitch Siebel™ Adapter Guide*
- *iWay SOAPswitch XML Adapter Guide*

Prerequisites

The following list describes prerequisites for using iWay SOAPswitch:

1. Database.

   You must install a database for use with iWay SOAPswitch.

   See iWay SOAPswitch Administration Guide

   You can also download MySQL version 4.1 for use with iWay SOAPswitch.

   See http://dev.mysql.com/downloads/mysql/4.1.html
2. Jetty server servlet.

   The iWay SOAPswitch product that ships with PeopleTools uses a standalone Jetty servlet server. You must download and install the Jetty servelet server version 5.1.4.

   Note. Running iWay SOAPswitch as a servlet under BEA WebLogic 8.1 or IBM WebSphere 5.1 is not yet supported.

   See http://jetty.mortbay.org/jetty/download.html

 Copyright © 1988-2007, Oracle. All rights reserved.

Starting and Stopping iWay SOAPswitch

This section provides discusses how to:

- Start iWay SOAPswitch Server.
- Stop iWay SOAPswitch Server.

Common Elements Used in This Section

In Windows, this icon appears on the system tray when SOAPswitch Server is running.

Starting iWay SOAPswitch Server

To start iWay SOAPswitch server run the following Java Archive (JAR) file:

```
java -jar start.jar
```

Stopping iWay SOAPswitch Server

To stop iWay SOAPswitch run the following JAR file:

```
java -jar stop.jar
```

Logging In to iWay SOAPswitch

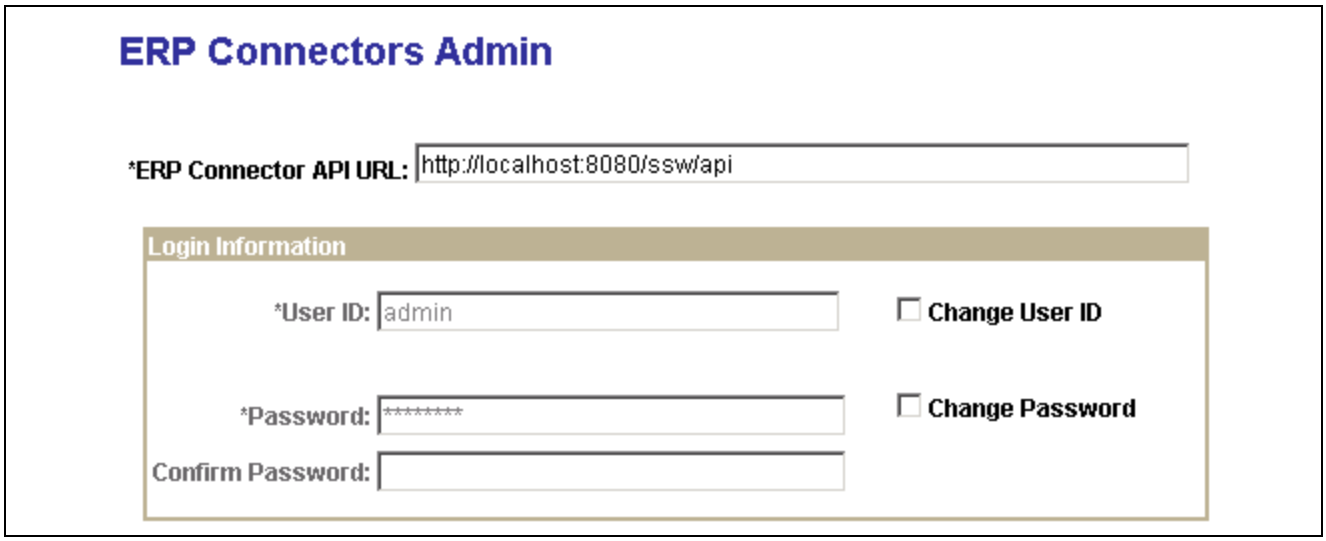
This section discusses how to:

- Log in to iWay SOAPswitch.
- Change the iWay SOAPswitch login user ID and password.

Logging In to iWay SOAPswitch

Login information for iWay SOAPswitch is stored on the ERP Connectors Admin page in the ERP Connectors Admin component (PT_IBWSDLADMIN). To access this page, select PeopleTools, Integration Broker, Web Services, ERP Connector Admin.

Copyright © 1988-2007, Oracle. All rights reserved.



ERP Connectors Admin page

This login provides access to the iWay SOAPswitch functionality in the ERP Connectors component.

By default, you are automatically logged into iWay SOAPswitch with the following default settings and values:

ERP Connector API URL	Specifies the URL where SOAPswitch is installed in the following format, where *8080* is the port number: `http://<machinename>:8080/ssw/api` Ports 8080 is the default port to communicate with iWay SOAPswitch.
User ID	The default user ID is *admin.*
Password	The default password is *password.*

Note. Change the default iWay SOAPswitch user ID and password as soon as possible.

Changing the iWay SOAPswitch Login User ID and Password

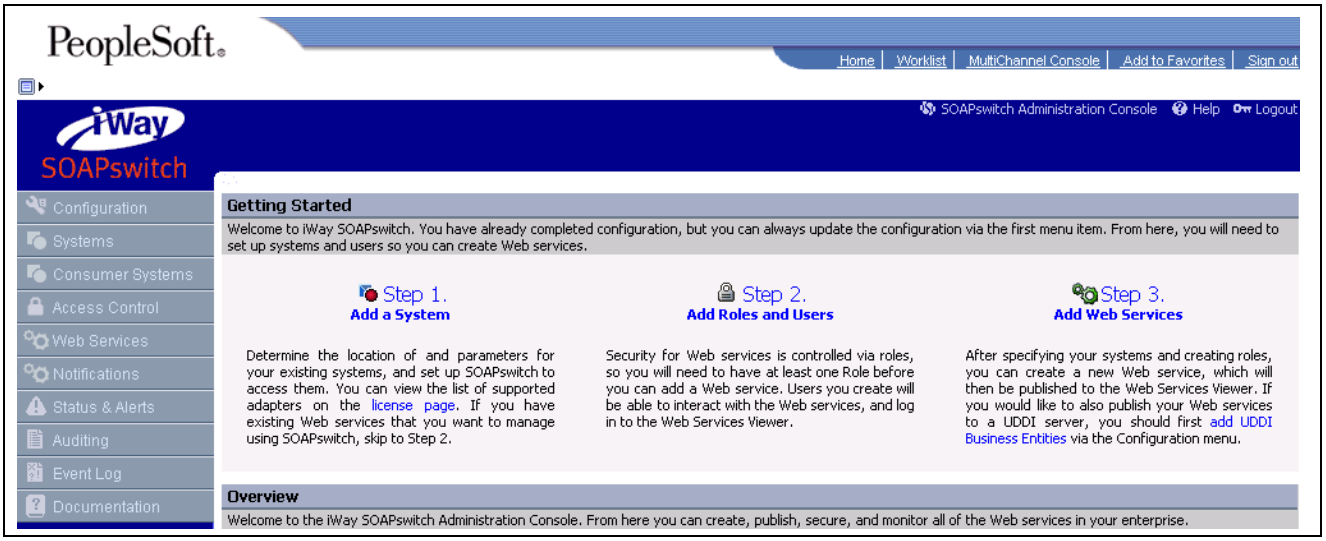
To change the iWay SOAPswitch login information:

1. To change the user ID:
   a. Select the Change User ID check box.
   b. In the User ID field, enter a new user ID.
2. To change the password:
   a. Select the Change Password check box.
   b. In the Password field, enter a new password.
   c. In the Confirm Password field, enter the new password again.
3. Click the Save button.

Copyright © 1988-2007, Oracle. All rights reserved.

Generating WSDL Using the ERP Connectors

You can generate WSDL from SAP, Oracle, and Siebel systems using the delivered ERP connectors that are accessible in the iWay SOAPswitch Administration Console. To access this page, select PeopleTools, Integration Broker, Web Services, ERP Connectors.



iWay SOAPswitch Administration Console

This section highlights the steps for generating outbound and inbound integrations using the ERP connectors. In addition, this section provides brief summaries of each task. Complete documentation for performing these tasks is located in the *iWay SOAPswitch Administration Guide* and the individual adapter guides for specific ERP environments. Information about how to access iWay SOAPswitch documentation is provided earlier in this chapter.

See Chapter 26, "Integrating with ERP Systems," iWay SOAPswitch and Adapter Documentation, page 573.

Steps for Generating WSDL for Outbound Transactions

To generate WSDL using the ERP connectors that are delivered with PeopleTools for outbound transactions from PeopleSoft systems:

1. Configure the appropriate ERP connector.
2. Add a system.
3. Add roles and users.
4. Add a web service to the system.

Steps for Generating WSDL for Inbound Transactions

To generate WSDL using the ERP connectors that are delivered with PeopleTools for inbound transactions to PeopleSoft systems.

1. Configure the appropriate ERP connector.
2. Add a system.
3. Add a consumer system.
4. Add roles and users.

Copyright © 1988-2007, Oracle. All rights reserved.

5. Add a destination.
6. Add a notification to the system.

Configuring ERP Connectors

iWay SOAPswitch provides ERP connectors for back-end systems that enable you to generate web services. When you generate a web service, WSDL is produced as a by-product. You can then import the WSDL into entities that can consume them, such as PeopleSoft systems.

You must configure an ERP connector for outbound and inbound transactions.

Please see the iWay SOAPswitch adapter documentation that is appropriate for your back-end system for setting up and configuring iWay SOAPswitch adapters.

Information about documentation and how to access it is provided earlier in this chapter.

See Chapter 26, "Integrating with ERP Systems," iWay SOAPswitch and Adapter Documentation, page 573.

Adding Systems

When you add a system, you use the System Definition Wizard to identify the back-end system to use to interact with iWay SOAPswitch.

You must add a system for outbound and inbound transactions.

To access the System Definition Wizard to add a system:

1. Access the iWay SOAPswitch Administration Console. Select PeopleTools, Integration Broker, Web Services, ERP Connectors.
2. Access the System Definition Wizard by performing one of the following actions:
   - Under the Getting Started section of the Administration Console, click Step 1. Add a System.
   - In the left navigation area, click Systems.

See Also

iWay SOAPswitch Administration Guide, "Chapter 4: Accessing Back-End Servers."

Adding Consumer Systems

When you add a consumer system, you specify the destination that iWay SOAPswitch uses for a notification. To add a consumer system, you use the Consumer System Definition Wizard to name the system and provide the URL to it.

Adding a consumer system is required for inbound transactions to PeopleSoft.

To access the Consumer System Definition Wizard to add a consumer system:

1. Access the iWay SOAPswitch Administration Console. Select PeopleTools, Integration Broker, Web Services, ERP Connectors.
2. In the left navigation bar, click Consumer Systems.

See Also

iWay SOAPswitch Administration Guide, "Chapter 7: Configuring Consumer System for Notification"

 Copyright © 1988-2007, Oracle. All rights reserved.

Adding Roles and Users

Security for web services is controlled through roles. Users that you create will be able to interact with the web services.

Adding roles and users is optional for outbound and inbound transactions.

To add users and roles, use the New Role Definition Wizard.

To access the New Role Definition Wizard to add users and roles:

1. Select PeopleTools, Integration Broker, Web Services, ERP Connectors.
2. Access the New Role Definition Wizard by performing one of the following actions:
   - Under the Getting Started section of the Administration Console, click Step 2. Add Roles and Users.
   - In the left navigation area, click Access Control.

See Also

iWay SOAPswitch Administration Guide, "Chapter 8: Using Access Control."

Creating Web Services

To create a web service from an SAP, Oracle, or Siebel back-end system, use the Web Services Definition Wizard.

When you create a web service, you make a connection with the back-end system and you can browse through the classes, methods, and other subcomponents of the back-end system, choose a service name, select the methods of the back-end system to expose, choose protocol and security mechanisms, and so on. The last step in the Web Service Definition Wizard is to publish the web service to PeopleSoft. A WSDL document is created as a by-product of publishing the web service.

When you create a web service, you connect in real time to SAP and Siebel systems to browse and choose data. On an Oracle system, you browse and choose data from static interface tables and connect to the system when you publish the service. If the Oracle system is not available when you attempt to publish the service, the system hangs and no error or warning message appears. However, a message is logged in the SOAPswitch error log.

When you create a web service from SAP in Linux environments, if you set SAP tracing to OFF, SAP continues to generate trace files if you cannot connect to SAP, which results in numerous trace files being generated on a failed connection. You have only to register a program (in this case iWay SOAPSwitch) if you want to perform an SAP outbound call, where SAP is the client and SOAPSwitch is the server. If you don't have such an SAP outbound scenario, just remove TOMMY from the SSW system settings (ProgramID).

Creating a web service is required for outbound transactions from PeopleSoft.

To add a web service, use the Web Services Definition Wizard.

To access the Web Services Definition Wizard:

1. Select PeopleTools, Integration Broker, Web Services, ERP Connectors.
2. Access the Web Services Definition Wizard by performing one of the following actions:
   - Under the Getting Started section of the Administration Console, click Step 3. Add Web Services.
   - In the left navigation area, click Web Services.

Copyright © 1988-2007, Oracle. All rights reserved.

After you publish the web service to PeopleSoft, in the Pure Internet Architecture the Consume Web Service wizard opens and you can consume the WSDL document into the PeopleSoft system.

See Chapter 24, "Consuming Services," page 533.

See Also

iWay SOAPswitch Administration Guide, "Chapter 12: Developing New Web Services."

iWay SOAPswitch Administration Guide, "Chapter 16: Maintaining Web Services."

Adding Destinations

When you add a destination, you specify the machine and URL where iWay SOAPswitch sends a notification. For inbound transactions, the destination is the PeopleSoft system.

When you create a destination definition, the name that you enter in the Name field must be the same as the machine name on which the integration gateway resides. For example:

```
machine051503
```

In addition, the URL that you enter in the URL field must be the same as the integration gateway URL that is specified on the Gateways page, with the exception that the URL that you enter in the destination definition is appended with the HTTP listening connector. For example, assume that the URL that is specified on the Gateways page is:

```
http://machine051503/PSIGW/PeopleSoftListeningConnector
```

Then the URL that you enter in the destination definition is:

```
http://machine051303/PSIGW/HTTPListeningConnector
```

Creating a destination is required for inbound transactions to PeopleSoft.

To create a destination, you use the Destination Definition Wizard.

To access the Destination Definition Wizard:

1. Access the iWay SOAPswitch Administration Console. Select PeopleTools, Integration Broker, Web Services, ERP Connectors.
2. In the left navigation area, click Notifications.
3. In the upper right corner of the page, select the Destinations tab. The Destination Definition Wizard appears.

See Also

iWay SOAPswitch Administration Guide, "Chapter 18: Managing Web Services for Notification," Create a Destination.

Creating Notifications

When you create a web service, you can browse through the classes, methods, and other subcomponents of a back-end system, choose a service name, select the methods of the back-end system to expose, choose protocol and security mechanisms, and so on.

Creating a notification is required for inbound transactions to PeopleSoft.

 Copyright © 1988-2007, Oracle. All rights reserved.

Creating Notifications to PeopleSoft

To create a notification:

1. Access the iWay SOAPswitch Administration Console. Select PeopleTools, Integration Broker, Web Services, ERP Connectors.
2. In the left navigation area, click Notifications.
3. In the upper right corner of the page, verify that the Notifications tab is selected.

See Also

iWay SOAPswitch Administration Guide, "Chapter 18: Managing Web Services for Notification."

Copyright © 1988-2007, Oracle. All rights reserved.

Copyright © 1988-2007, Oracle. All rights reserved.

CHAPTER 27

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.

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.

Copyright © 1988-2007, Oracle. All rights reserved.

WS-Security

Web services security (WS-Security) is implemented on the integration gateway for inbound and outbound integrations with third-party systems.

UsernameToken as a means 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.

WS-Security adds a layer of security to sending and receiving service operations by adding a UsernameToken that identifies the sender and authenticate its identity to the web service provider. .

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.

You can implement WS-Security with integration partners running on PeopleSoft 8.48 and later systems and third-party systems.

See Also

Enterprise PeopleTools 8.49 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.

 Copyright © 1988-2007, Oracle. All rights reserved.

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

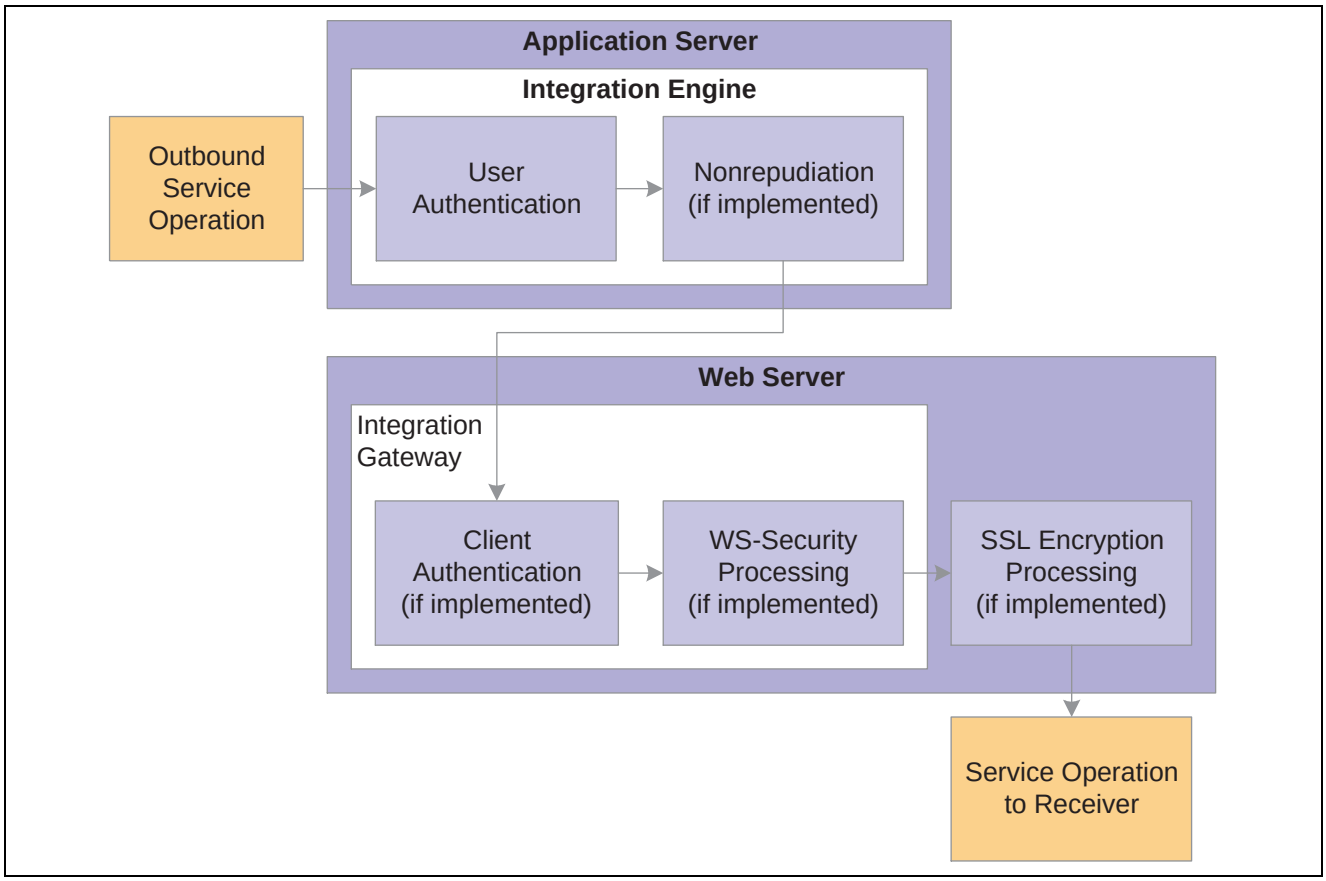
This section discusses:

- 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:

Copyright © 1988-2007, Oracle. All rights reserved.



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.

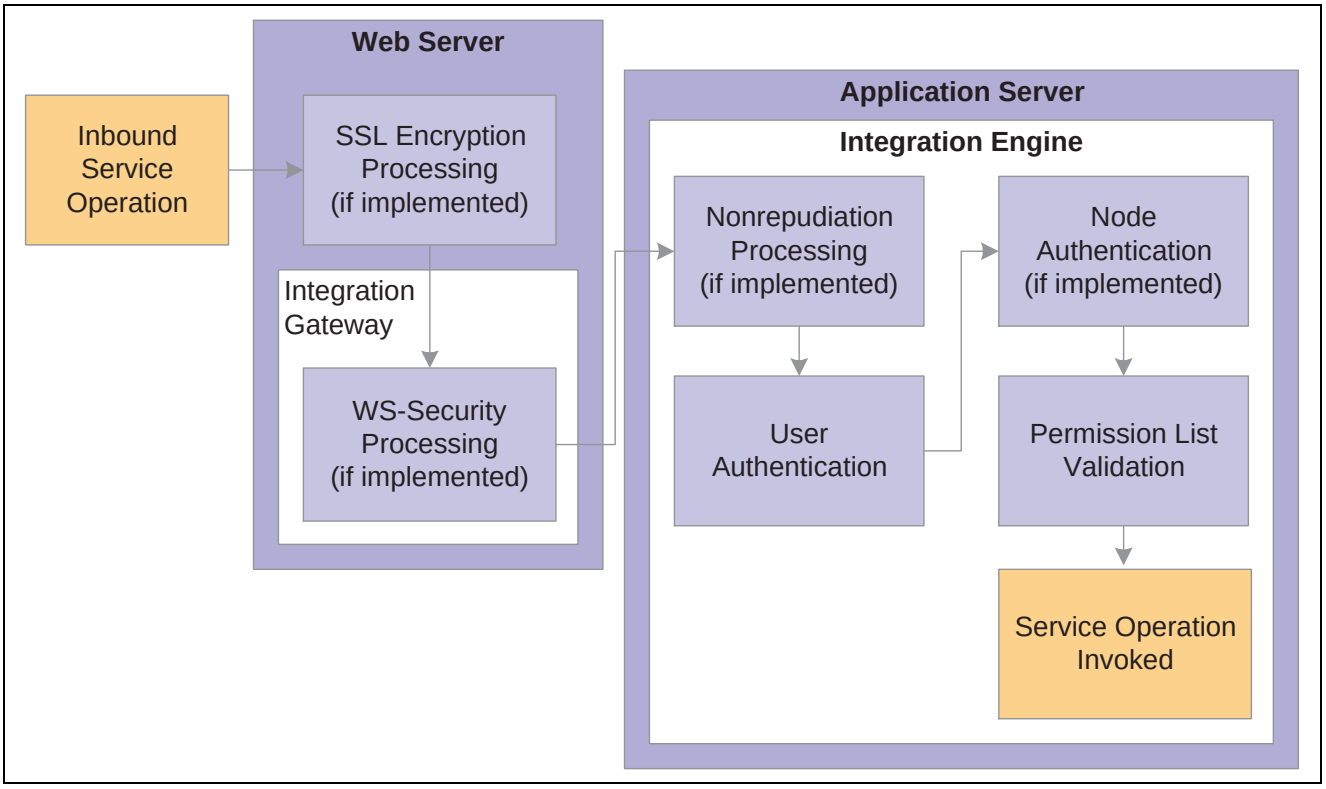
WS-Security Outbound WS-Security processing includes generating the UsernameToken for the WS-Security SOAP header. This process may also involve encrypting and digitally signing the data, if specified in the WS-Security parameters on the node.

SSL encryption SSL encryption on outbound integrations establishes a secure web server connection with an integration partner.

Copyright © 1988-2007, Oracle. All rights reserved.

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.

Copyright © 1988-2007, Oracle. All rights reserved.

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.

 Copyright © 1988-2007, Oracle. All rights reserved.

- 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*.

Copyright © 1988-2007, Oracle. All rights reserved.

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

This is a document that contains the entity's public key. The CSR is typically generated in Privacy Enhanced Mail (PEM) format, which is base 64–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.

Copyright © 1988-2007, Oracle. All rights reserved.

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.

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.

Copyright © 1988-2007, Oracle. All rights reserved.

Pages Used to Install Application Server-Based Digital Certificates

Page Name	Object Name	Navigation	Usage
Digital Certificates page	ADMINISTER_CERTS	Select PeopleTools, Security, Security Objects, Digital Certificates.	Use this page to: • Install root certificates. • Install signed public key certificates. • Install a remote certificate. • Export a certificate.
Request New Certificate	CERT_REQ_SBP	On the Digital Certificates page, add a new row for the local node and click the Request link.	Obtain and import the a local node certificate:

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 dropdown 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 -----).

Copyright © 1988-2007, Oracle. All rights reserved.

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.

Copyright © 1988-2007, Oracle. All rights reserved.

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 27, "Setting Up Secure Integration Environments," Accessing Certificate Properties, page 596.

 Copyright © 1988-2007, Oracle. All rights reserved.

- 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 27, "Setting Up Secure Integration Environments," Exporting and Converting Certificates, page 597.

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.

Install 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.

Copyright © 1988-2007, Oracle. All rights reserved.

- 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:

 Copyright © 1988-2007, Oracle. All rights reserved.

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 27, "Setting Up Secure Integration Environments," Accessing Certificate Properties, page 596.

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.

Copyright © 1988-2007, Oracle. All rights reserved.

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 <PS_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.

 Copyright © 1988-2007, Oracle. All rights reserved.

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 alone, 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 <PS_HOME>\webserv\<DOMAIN>\peoplesoft\applications\PSIGW\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 <PS_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:

Copyright © 1988-2007, Oracle. All rights reserved.

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:

 Copyright © 1988-2007, Oracle. All rights reserved.

```
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 <PS_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 <PS_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"
   ```

Copyright © 1988-2007, Oracle. All rights reserved.

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 <PS_HOME>\webserv\<DOMAIN>\peoplesoft\applications\PSIGW\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:/<PS_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.

 Copyright © 1988-2007, Oracle. All rights reserved.

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 7, "Managing Integration Gateways," Encrypting Passwords Using the PSCipher Java Utility, page 66.

2. Access the wss.properties file.

   The location is <PS_HOME>\webserv\<DOMAIN>\peoplesoft\applications\PSIGW\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 BEA WebLogic web servers.
- Install digital certificates on IBM WebSphere web servers.
- Install digital certificates on Oracle Application Server 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 BEA WebLogic, IBM WebSphere, or Oracle Application Server 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.

Copyright © 1988-2007, Oracle. All rights reserved.

The information in section outlines the basic steps required to obtain and install the certificates and keys that you need. BEA WebLogic, IBM WebSphere and Oracle Application Server each 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 on BEA WebLogic

This section describes how to install digital certificates for SSL encryption for the BEA 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 BEA 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:

   ```
   <PS_HOME>\webserv\peoplesoft\keystore
   ```

2. Open PSKeyManager.
   a. Open a command prompt and navigate to <PS_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.

 Copyright © 1988-2007, Oracle. All rights reserved.

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 BEA WebLogic:

1. Start PSKeyManager.
    a. Open a command prompt and navigate to <PS_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.

Copyright © 1988-2007, Oracle. All rights reserved.

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 <PS_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 <PS_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.

 Copyright © 1988-2007, Oracle. All rights reserved.

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 *PT849GATEWAY*.
Common name for this certificate.	Enter a name, such as *PT849GATEWAY*.

Setting Up BEA WebLogic for SSL Encryption

To set up BEA 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 BEA 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.

Copyright © 1988-2007, Oracle. All rights reserved.

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 dropdown list, select *Client Certs Requested and Enforced*.
12. Click the Apply button.
13. Restart the web server.

Installing Digital Certificates 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:

   ```
   PS_HOME\webserv\<cell_name>_<node_name>_<server_name>\peoplesoft.ear\keystore\pskey
   ```

2. Click the Download button and load the file to <PS_HOME>\webserv\<DOMAIN>. For example:

   ```
   <PS_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.

 Copyright © 1988-2007, Oracle. All rights reserved.

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:

   ```
   <PS_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.

Copyright © 1988-2007, Oracle. All rights reserved.

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 *PT849GATEWAY*.
Common name for this certificate.	Enter a name, such as *PT849GATEWAY*.

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.

 Copyright © 1988-2007, Oracle. All rights reserved.

- 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:

      ```
      <PS_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.

Copyright © 1988-2007, Oracle. All rights reserved.

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.

Installing Digital Certificates for SSL Encryption on Oracle Application Server

This section discusses how to:

- Create a Wallet repository in Oracle Wallet Manager.
- Generate and import public keys
- Generate Private Keys and CSRs.
- Submit CSRs to CAs for signing.
- Import signed private keys into Oracle Wallet Manager.
- Set up gateway private keys.
- Set up Oracle Application Server for SSL encryption.

Creating Wallet Repositories in Oracle Wallet Manager

A Wallet is a repository in which to securely store user certificates and the trust points needed to validate the certificates of peers.

The steps in this section are based on Oracle Wallet Manager 10.1.0.2.0.

To create a wallet:

1. Launch Oracle Wallet Manager.
   - If running on Windows, select Start, Programs, Oracle - <OAS_HOME>, Integrated Management Tools, Wallet Manager
   - If running on Unix, at the command line, enter *<OAS_HOME>/bin/owm*.

   Oracle Wallet Manager appears.

2. From the Wallet menu, select *New*.

   A New Wallet dialog box appears.

3. In the Wallet Password field, enter create a password. This password protects the unauthorized use of your credentials.

   The password must be a minimum of eight characters in length, and contain alphabetic characters combined with numbers or special characters.

4. In the Confirm Password field, re-enter your password.
5. From the Wallet Type dropdown list box, select *Standard*.

 Copyright © 1988-2007, Oracle. All rights reserved.

6. Click the OK button.

   An alert displays and informs you that a new empty wallet has been created and prompts you to add a certificate request.

7. Click the No button.

   The Oracle Wallet Manager main window appears. The new wallet you created appears in the left window pane. The certificate has a status of *[Empty]*, and the wallet displays its default trusted certificates.

8. Save the new wallet.

   a. From the Wallet Menu, select *Save As.*

   b. Browse to the following location: ORACLE_HOME\Apache\Apache\conf\ssl.wlt.

   Note. This location must be used in the SSL configuration for clients and servers.

   c. Enter a name for the wallet.

   d. Click the OK button.

9. From the Wallet menu, select check the Auto Login box.

10. From the Wallet menu, select Save.

11. Open the file <OAS_HOME>\Apache\Apache\conf\ssl.conf and modify the line SSLWallet file:OAS_NAME\Apache\Apache\conf\ssl.wlt\default to SSLWallet file:OAS_NAME\Apache\Apache\conf\ssl.wlt\<*mywallet*>, where <mywallet> is the name of the wallet just created.

Generating and Importing Public Keys (OAS)

Before you can generate and import public keys into the PeopleSoft system, 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. Launch Oracle Wallet Manager.
2. From the Operations menu, select Import Trusted Certificate.

   An Import Trusted Certificate dialog box appears.

3. Select the option *Select a file that contains the certificate* and click the OK button.
4. Browse to the location where the root CA was downloaded, select it and click the Open button.

   Oracle Wallet Manager appears and displays the imported CA certificate.

5. Save the wallet.

Generating Private Keys and CSRs (OAS)

To generate a certificate request:

1. Launch Oracle Wallet Manager.
2. From the Operations menu, select Add Certificate Request.

   The Certificate Request window appears.

3. Complete the certificate request:

Copyright © 1988-2007, Oracle. All rights reserved.

Common Name	Enter the name of your machine. For example: *ple-machine1.peoplesoft.com*.
Organizational Unit	Enter the name of your organizational unit.
Organization	Enter the name of your company or organization.
Locality/City	Enter the locale or city where your organizational unit or company is located.
State/Province	Enter the state or province where your organizational unit or company is located.

4. Click the OK button.

   A message informs you that a certificate request was successfully created. You can either copy the certificate request text from the body of the dialog box and paste it into an email message to send to a certificate authority, or you can export the certificate request to a file.

5. Click the OK button.

   The Oracle Wallet Manager main window appears and the status of the certificate changes to *[Requested]*.

Submitting CSRs to CAs for Signing (OAS)

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 Oracle Wallet Manager.

To access the CSR:

To submit the certificate request to the CA to obtain a public key:

1. Launch Oracle Wallet Manager.
2. In the left navigation pane, click *Certificate:[Requested]*.

   Information about the certificate request displays in the right pane.

3. In the right pane, copy the certificate request to the clipboard using CTRL+V.

Importing Signed Private Keys into the Oracle Wallet Manager (OAS)

When your CA returns the signed certificate, you must import it into Oracle Wallet Manager.

To import the signed private key into the Wallet repository:

1. Launch Oracle Wallet Manager.
2. From the Operations menu, select Import User Certificate.

 Copyright © 1988-2007, Oracle. All rights reserved.

3. Select the option *Select a file that contains the certificate* and click the OK button.
4. Browse to the location where the signed certificate was downloaded, select it, and click the Open button.

   The Oracle Wallet Manager main window appears and in the left navigation pane, the status of the certificate changes to *[Ready]*.
5. Save the wallet

Setting Up Gateway Private Keys (OAS)

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 *PT849GATEWAY*.
Common name for this certificate.	Enter a name, such as *PT849GATEWAY.*

Setting Up Oracle Application Server for SSL Encryption

To set up Oracle Application Server for SSL encryption:

1. Edit the <OAS_HOME>/opmn/conf/opmn.xml file as follows:
   a. In a text editor open the file <OAS_HOME>/opmn/conf/opmn.xml.
   b. In the <ias-component> entry, change the start mode from *"ssl-disabled"* to *"ssl-enabled"*. The following example shows how the entry should look after the modification: *<data value="ssl-enabled"/>*
   c. Save the file.
2. Modify the <OAS_HOME>/Apache/Apache/conf/mod_oc4j.conf file.
   a. In a text editor open the file <OAS_HOME>/Apache/Apache/conf/mod_oc4j.conf file.
   b. Locate the section*<IfDefine SSL>*. turn on SSLEngine and update the in this file and do the followin
   c. Turn on SSLEngine.
   d. Update the location to the proper wallet location. For example: <OAS_HOME>\Apache\Apache\conf \ssl.wlt\mywallet
   e. Save the file
3. Modify the <OAS_HOME>/Apache/Apache/conf/ssl.conf file.
   a. In a text editor open the file OAS_HOME/Apache/Apache/conf/ssl.conf.
   b. Verify that the SSLEngine is on and update the SSLWallet path to point to proper wallet location
   c. Save the file.
4. Update the distributed cluster management database with the change: ORACLE_HOME/dcm/bin /dcmctl updateconfig -ct opmn
5. Reload OPMN using the following command: OAS_HOME/opmn/bin/opmnctl reload

Copyright © 1988-2007, Oracle. All rights reserved.

6. Run <OAS_HOME>/dcm/bin/dcmctl updateconfig.
7. Stop Oracle HTTP Server using the Application Server Control Console or using one of the following commands:
   - For Windows:< OAS_HOME>\opmn\bin> opmnctl [verbose] stopproc ias-component=HTTP_Server
   - For Unix: <OAS_HOME>/opmn/bin> opmnctl [verbose] stopproc ias-component=HTTP_Server
8. Start Oracle HTTP Server using Application Server Control Console or using one of the following commands:
   - For Windows: <OAS_HOME>\opmn\bin> opmnctl [verbose] startproc ias-component=HTTP_Server
   - For Unix: <OAS_HOME>/opmn/bin> opmnctl [verbose] startproc ias-component=HTTP_Server
9. Verify that SSL was enabled successfully by navigating to the SSL port, for example: *https://hostname:4443*.

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

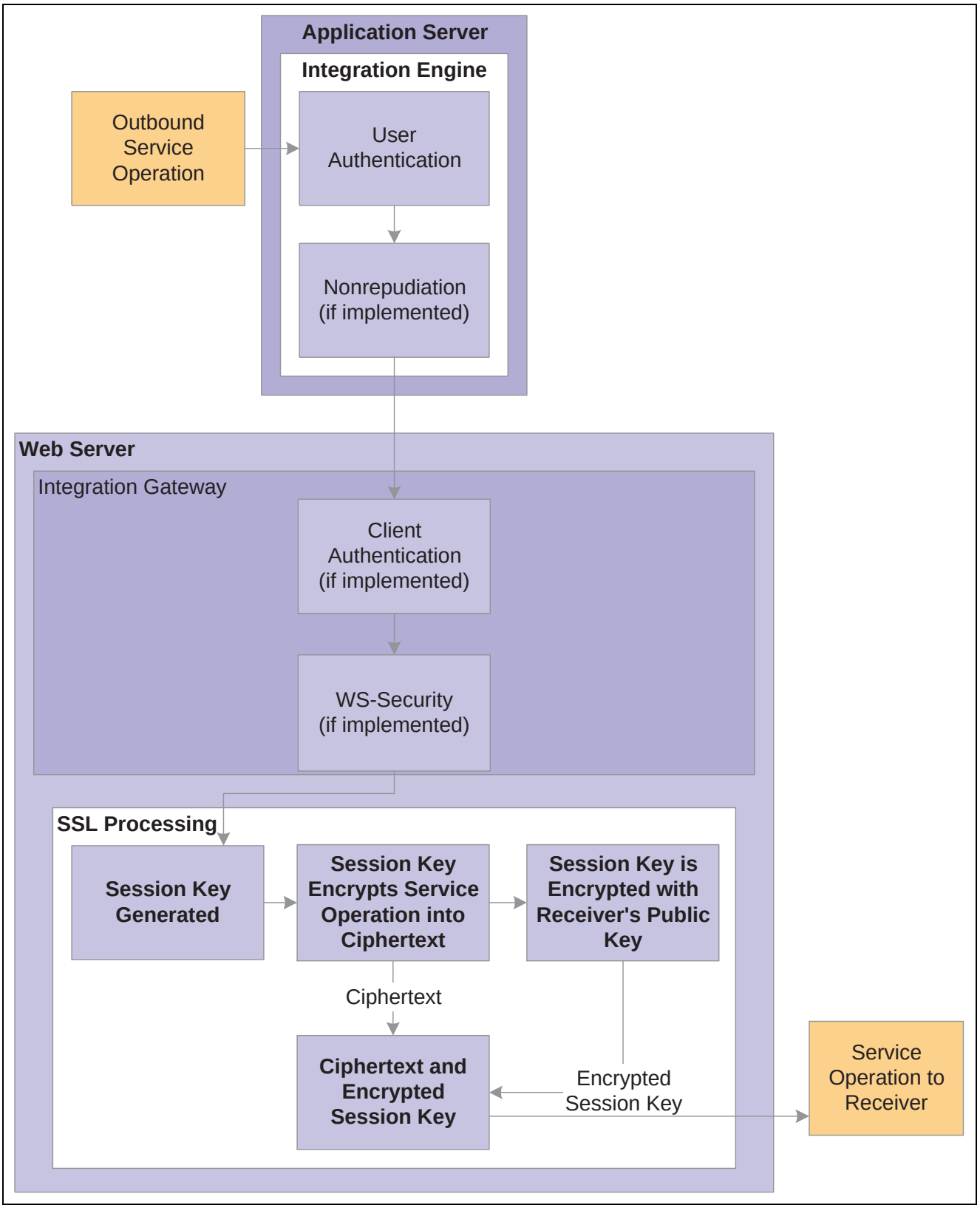
This section discusses:

- 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:

 Copyright © 1988-2007, Oracle. All rights reserved.



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.

Copyright © 1988-2007, Oracle. All rights reserved.

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.

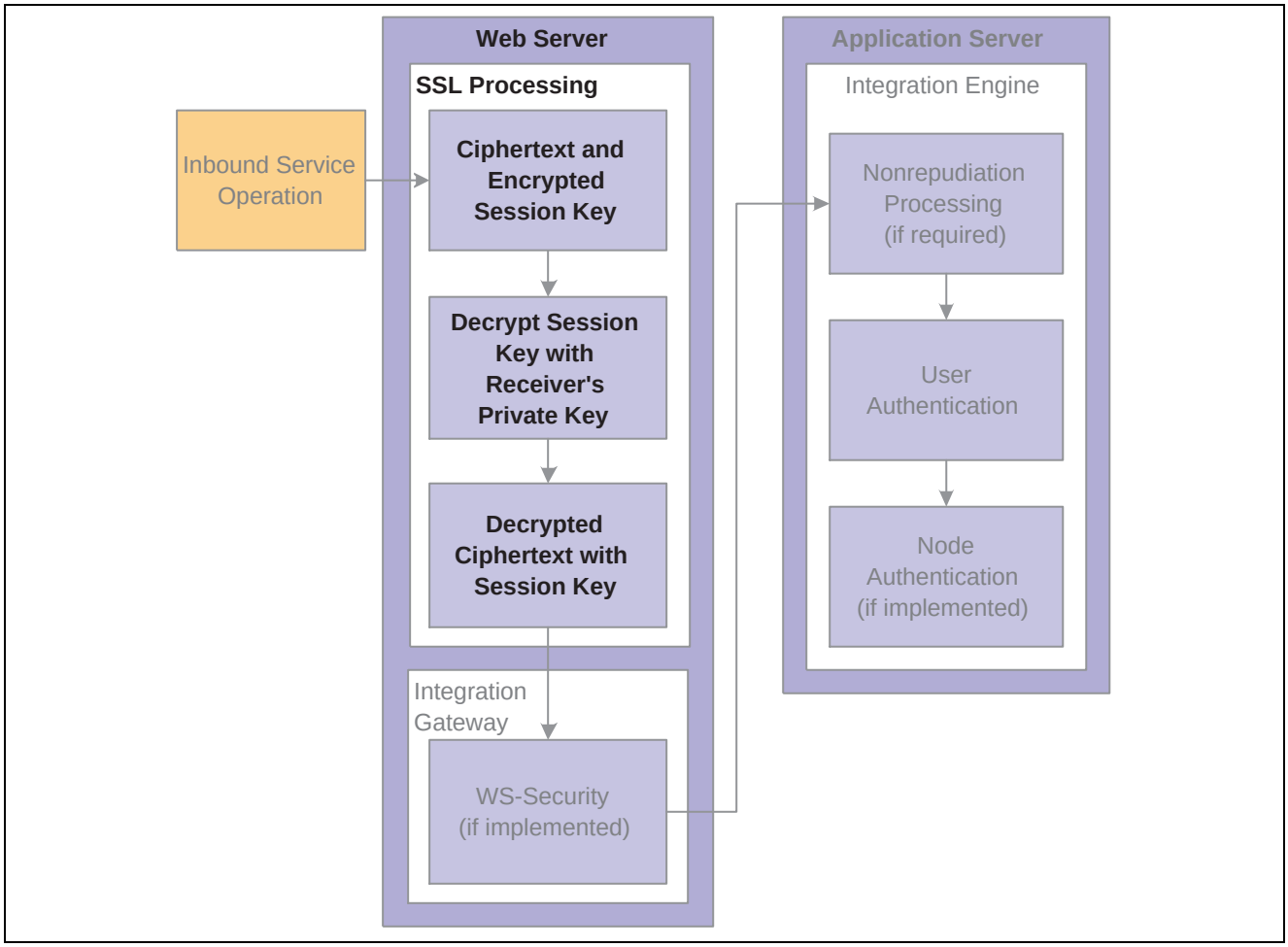
Your web server generates a session key to encrypt the plain text outbound request contents into ciphertext. Then the web server encrypts the ciphertext and session key using your integration partner's public key that was sent to you in the digital certificate.

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.

Copyright © 1988-2007, Oracle. All rights reserved.

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 7, "Managing Integration Gateways," Using the integrationGateway.properties File, page 64.

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 WS-Security

This section discusses how to:

- Install digital certificates for inbound integrations.
- Install digital certificates for outbound integrations.
- Set up WS-Security for inbound integrations.

Copyright © 1988-2007, Oracle. All rights reserved.

- Set up WS-Security for outbound integrations.
- Configure WS-Security for outbound integrations.
- 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 the Oasis Standard 1.0 WS-Security schema, which conforms to the Web Service Security standard version 1.

See http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd

Within this framework, PeopleSoft implements:

- Username Token Profile 1.0
- X.509 Token Profile 1.0

PeopleSoft's XML signature and XML encryption feature support surrounds the UsernameToken profile. As a result, XML signature and XML encryption are fully functional for the UsernameToken section of the SOAP header, but not necessarily for the entire XML SOAP message.

The PeopleSoft implementation of WS-Security supports:

- Clear-text UsernameToken. (Password is optional.)
- Digitally signed UsernameToken
- Digitally signed UsernameToken.

  Digital signatures apply to the SOAP message header and SOAP message body.
- Only the encrypted WS UsernameToken is included in the SOAP header

The following Oasis schema namespaces are not supported:

- WSSE_NS_OASIS_2002_07: "http://schemas.xmlsoap.org/ws/2002/07/secext"
- WSSE_NS_OASIS_2002_12: "http://schemas.xmlsoap.org/ws/2002/12/secext"
- WSSE_NS_OASIS_2003_06: "http://schemas.xmlsoap.org/ws/2003/06/secext"

UsernameToken Credentials

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 credentials are 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 credentials 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.

 Copyright © 1988-2007, Oracle. All rights reserved.

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.

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>
```

Understanding WS-Security Processing in PeopleSoft Integration Broker

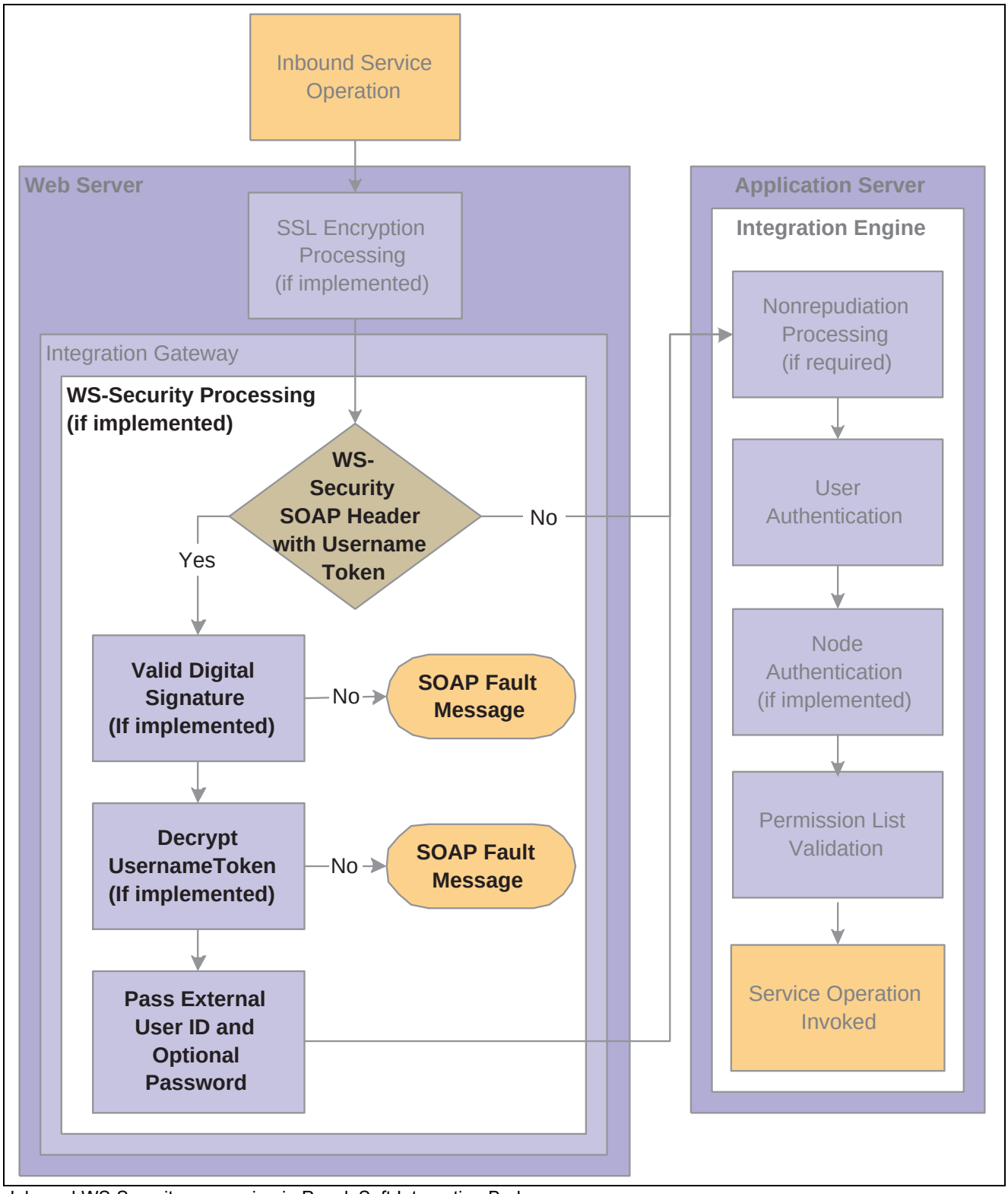
This section provides overviews of:

- Inbound WS-Security processing.
- Outbound WS-Security processing.

Inbound WS-Security Processing in PeopleSoft Integration Broker

The inbound processing of service operations that are WS-Security compliant occurs on the integration gateway. The following diagram illustrates inbound WS-Security processing in PeopleSoft Integration Broker:

Copyright © 1988-2007, Oracle. All rights reserved.



Inbound WS-Security processing in PeopleSoft Integration Broker

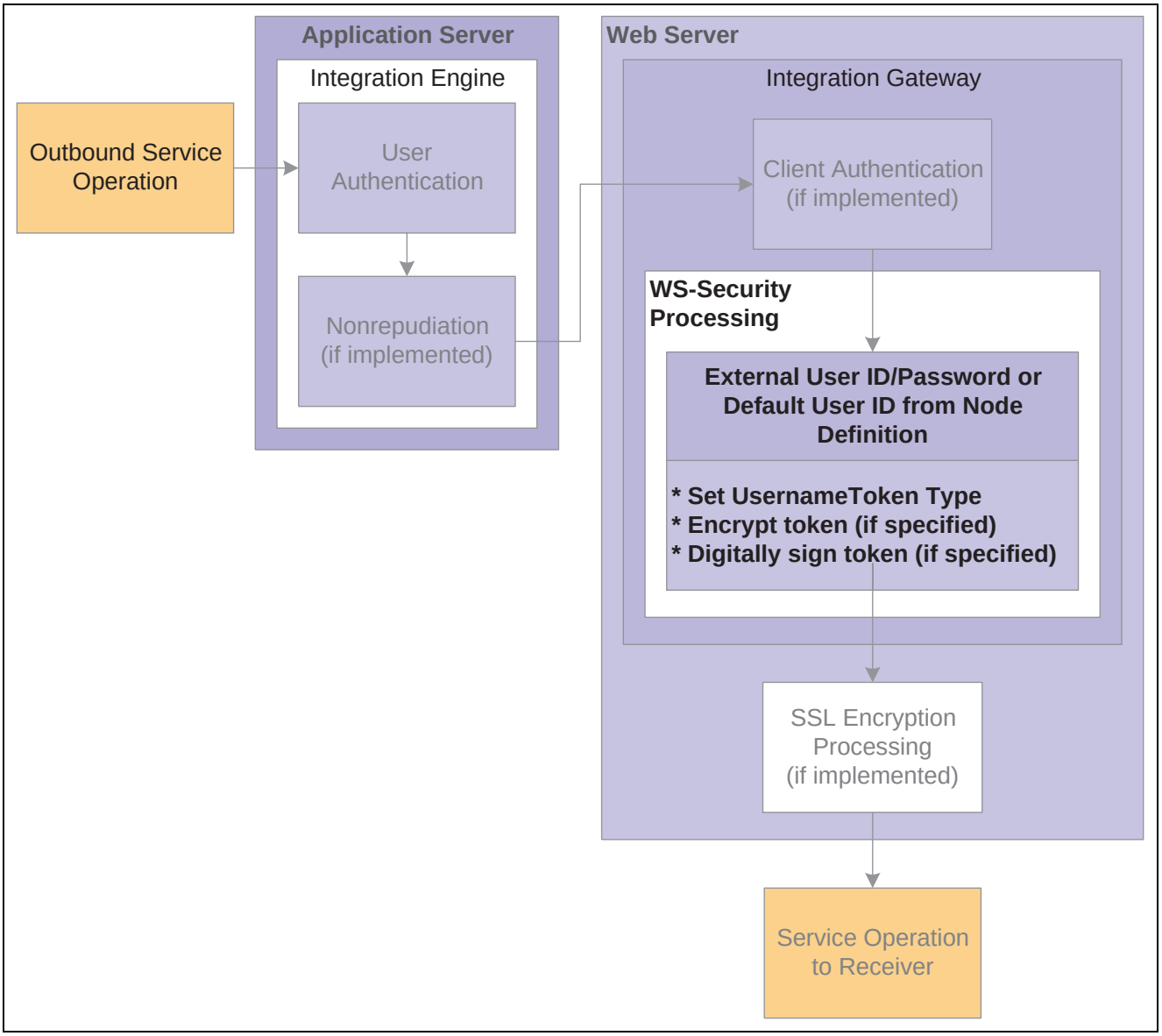
When any transaction arrives at the integration gateway, the PeopleSoft system checks for the existence of a WS-Security SOAP header. If it exists, the integration gateway validates the digital signature if it exists, and decrypts the UsernameToken and optional password to restore the user ID information to clear text format.

Copyright © 1988-2007, Oracle. All rights reserved.

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 in PeopleSoft Integration Broker

The following diagram illustrates WS-Security processing by PeopleSoft Integration Broker on outbound integrations:



Outbound WS-Security processing in PeopleSoft Integration Broker

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 User ID*, *External Password*, or *Default User ID*. Additionally, you can choose to encrypt and digitally sign the UsernameToken credentials.

See Chapter 27, "Setting Up Secure Integration Environments," Implementing WS-Security for Outbound Integrations, page 624 and Chapter 27, "Setting Up Secure Integration Environments," Describing WS-Security Configuration Options for Outbound Integrations, page 626.

Copyright © 1988-2007, Oracle. All rights reserved.

Prerequisites for Implementing WS-Security in PeopleSoft Integration Broker

To implement WS-Security in PeopleSoft Integration Broker you must install digital certificates.

See Chapter 27, "Setting Up Secure Integration Environments," Installing Integration Gateway-Based Digital Certificates, page 598.

Implementing WS-Security for Inbound Integrations

There is no set up required for implementing WS-Security on inbound integrations. The integration gateway handles all inbound processing.

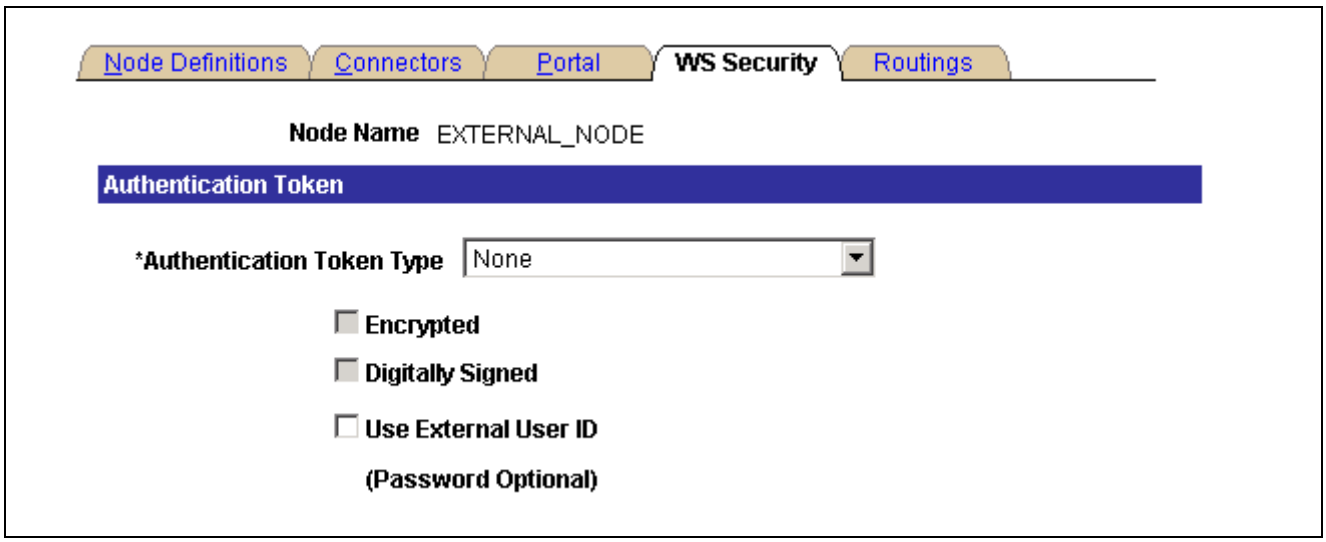
Implementing WS-Security for Outbound Integrations

This section discusses how to:

- Specify an authentication token.
- 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:



Nodes–WS-Security page

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.

Copyright © 1988-2007, Oracle. All rights reserved.

4. From the Authentication Token Type dropdown list box select *Username Token*.

   The Encrypted and Digitally Signed options become enabled.

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 27, "Setting Up Secure Integration Environments," Describing WS-Security Configuration Options for Outbound Integrations, page 626.

   Encrypt (Optional.) Check the box to encrypt the UsernameToken information, including the username and password.

   Digitally Signed (Optional.) Check the box to digitally sign the UsernameToken information, including the username and password.

   Use External User ID (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.

6. Click the Save button.
7. Click the Node Definitions tab.

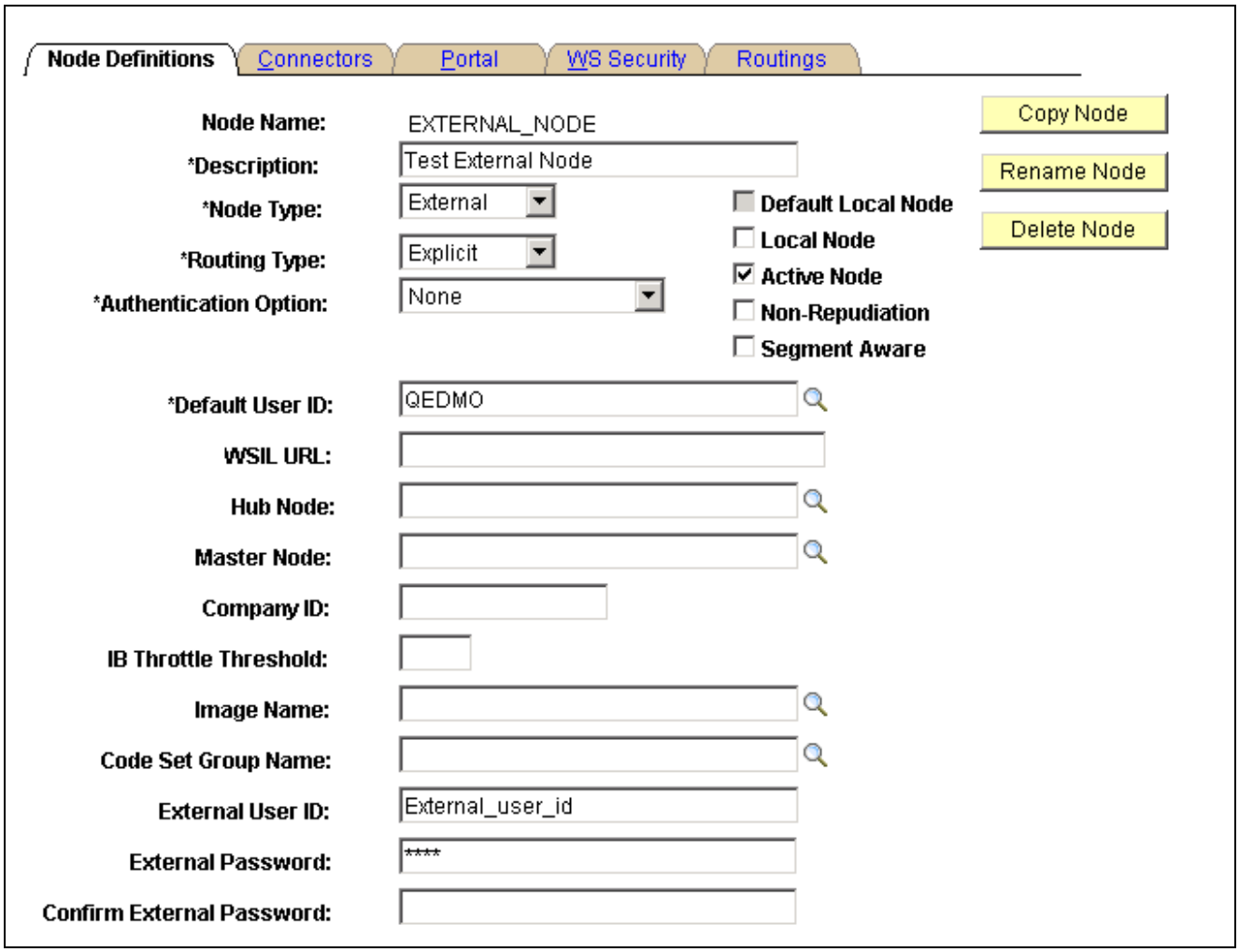
   The Node Definitions page appears.

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 next section.

Specifying External User IDs and Passwords

If when choosing the authentication token type on the WS-Security page you chose the option Use and External User ID, you must specify an external user ID on the Node Definitions page. The following example shows the Node Definitions page:

Copyright © 1988-2007, Oracle. All rights reserved.



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.

Describing WS-Security Configuration Options for Outbound Integrations

This section discusses:

- Recommended WS-Security configurations for outbound integrations.

Copyright © 1988-2007, Oracle. All rights reserved.

- Supported WS-Security configurations for outbound integrations.
- Non-secure WS-Security configurations for outbound integrations.

Recommended WS-Security Configurations for Outbound Integrations

The following table highlights recommended WS-Security configurations on the PeopleSoft system for outbound integrations. 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 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

The following table highlights supported WS-Security configurations on the PeopleSoft system for outbound integrations. Note that the configuration is always performed on the remote node and that the node type is always *External*.

Copyright © 1988-2007, Oracle. All rights reserved.

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

The following table highlights non-secure WS-Security configurations on the PeopleSoft system for outbound integrations. 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

This section provides the following WS-Security code examples:

Copyright © 1988-2007, Oracle. All rights reserved.

- 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/
```

Copyright © 1988-2007, Oracle. All rights reserved.

```
                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>174332155640842765207620
                  </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
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
                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>
```

Copyright © 1988-2007, Oracle. All rights reserved.

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 27, "Setting Up Secure Integration Environments," Installing Integration Gateway-Based Digital Certificates, page 598

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.

Process 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.

 Copyright © 1988-2007, Oracle. All rights reserved.

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.

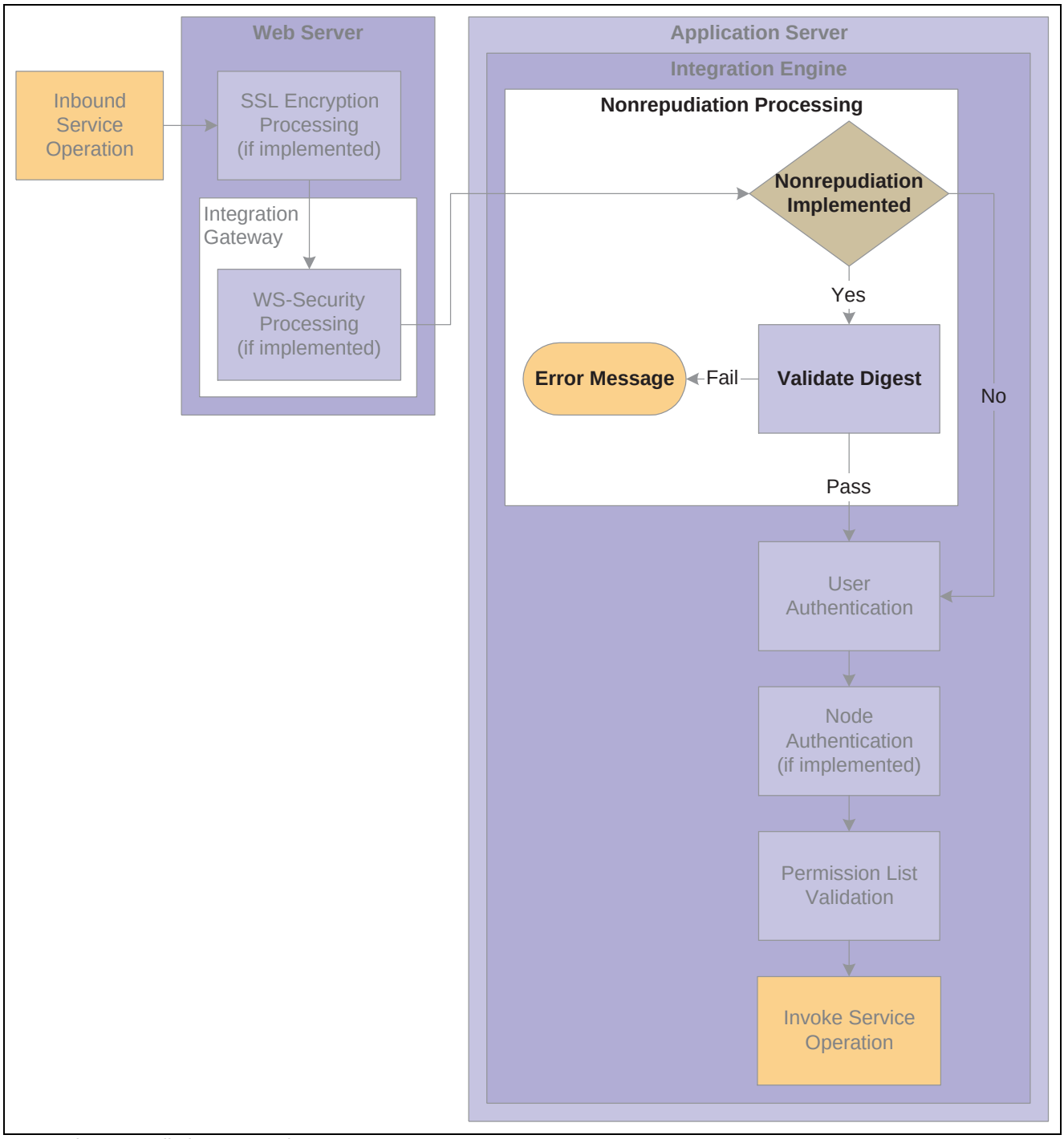
Nonrepudiation produces the following results:

- 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:

Copyright © 1988-2007, Oracle. All rights reserved.



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.

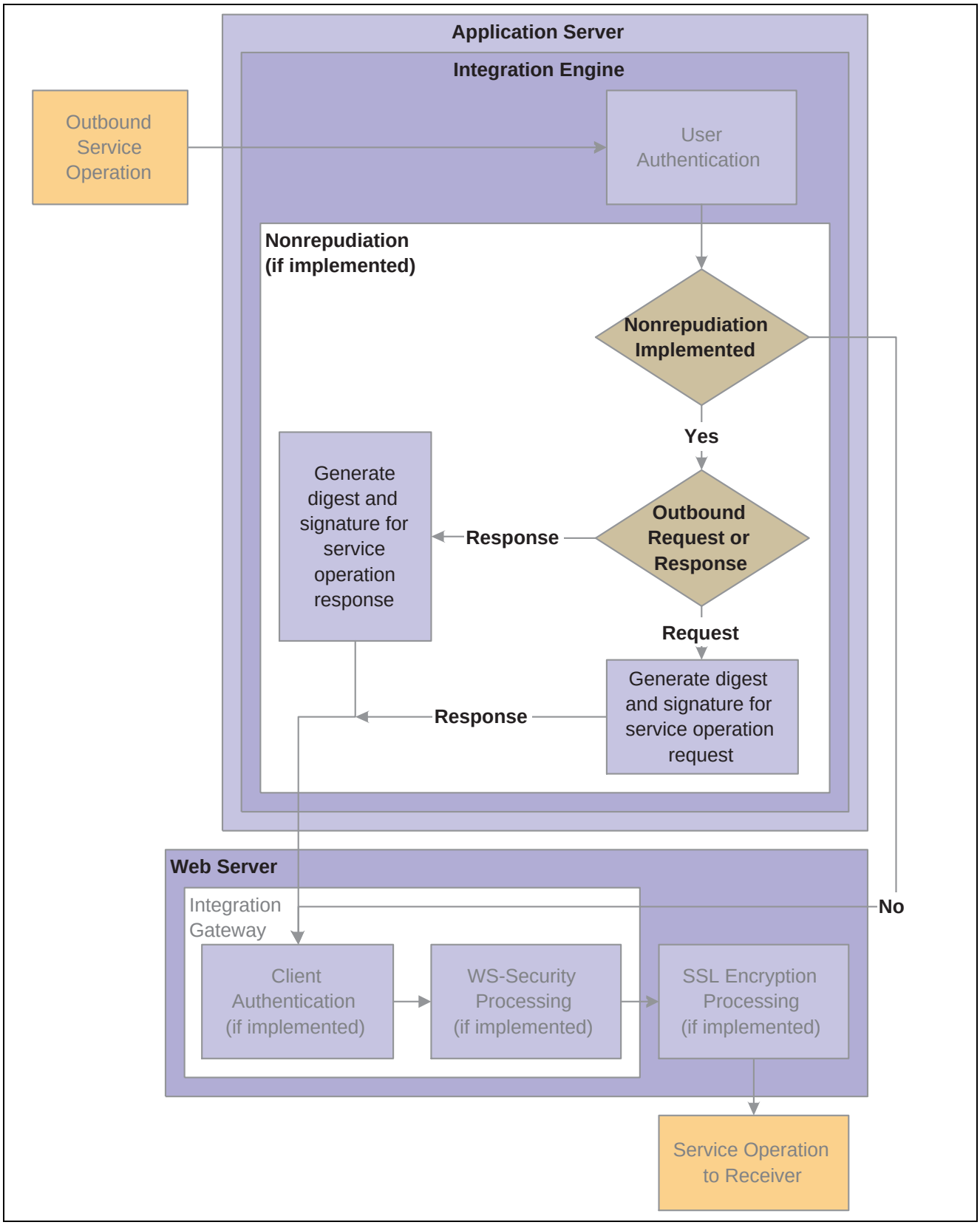
If the system is able to validate the integrity of the digest and the service operation content, the service operation then goes through the user authentication process. If the system is unable to validate the digest or the service operation content, the transaction fails.

Copyright © 1988-2007, Oracle. All rights reserved.

Outbound Nonrepudiation Processing

The following diagram illustrates outbound nonrepudiation processing:

Copyright © 1988-2007, Oracle. All rights reserved.



Outbound nonrepudiation processing

On outbound service operations, the system determines if the service operation is a request or a response.

Copyright © 1988-2007, Oracle. All rights reserved.

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 to implement nonrepudiation.

See Chapter 27, "Setting Up Secure Integration Environments," Installing Application Server-Based Digital Certificates, page 591.

Configuring Nonrepudiation

Nonrepudiation functionality requires the following tasks:

- You must supply the digital certificates containing the private and public keys required for nonrepudiation transactions.

  These elements are required at every node that participates in a nonrepudiation transaction; PeopleSoft Integration Broker handles the mechanics of applying the keys.

- You must select the Non-Repudiation check box on the service operation to indicate that nonrepudiation is to be implemented for the operation.

  When you do this on one system, you must select this option on the same service operation on every system that will participate in transactions using the service operation.

- You must select the Non-Repudiation check box in a node definition to indicate that it supports nonrepudiation.

  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.

- To save nonrepudiation service operations for future reference, you must make sure that they are archived.

  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.

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.

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.

Understanding User Authentication

In PeopleTools 8.48 and later releases, access to invoke service operations is enforced at the user level.

Copyright © 1988-2007, Oracle. All rights reserved.

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.

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

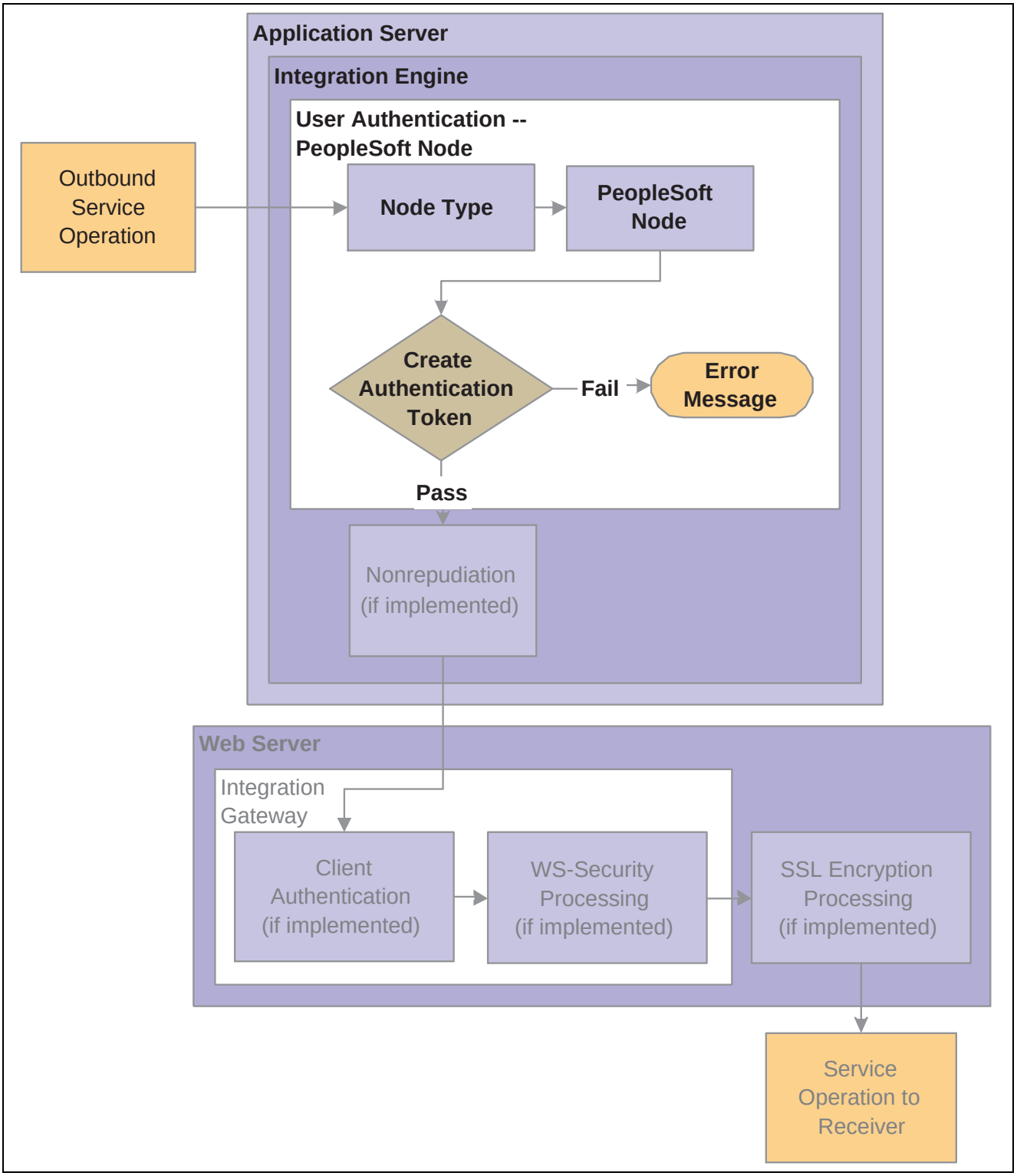
The outbound user authentication process determines the user ID to identify and attach to the outbound service operation. If the receiving system is a PeopleSoft system, the system validates the user ID and if the user ID belongs to the permission list to which the service operation is assigned, the 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.

 Copyright © 1988-2007, Oracle. All rights reserved.

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 — PeopleSoft Node

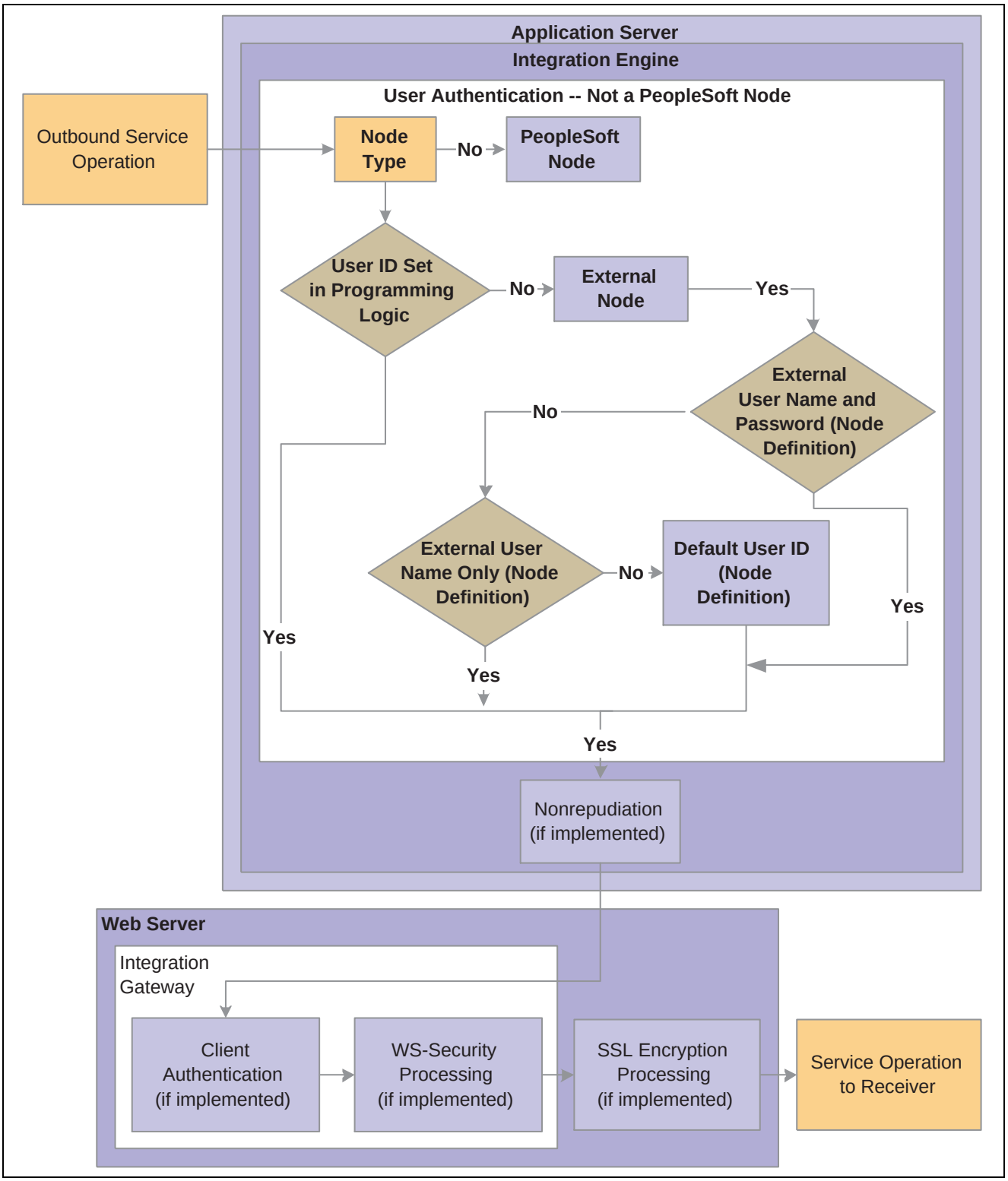
Copyright © 1988-2007, Oracle. All rights reserved.

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 as a trusted node.

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:

 Copyright © 1988-2007, Oracle. All rights reserved.



Outbound user authentication-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.

Copyright © 1988-2007, Oracle. All rights reserved.

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

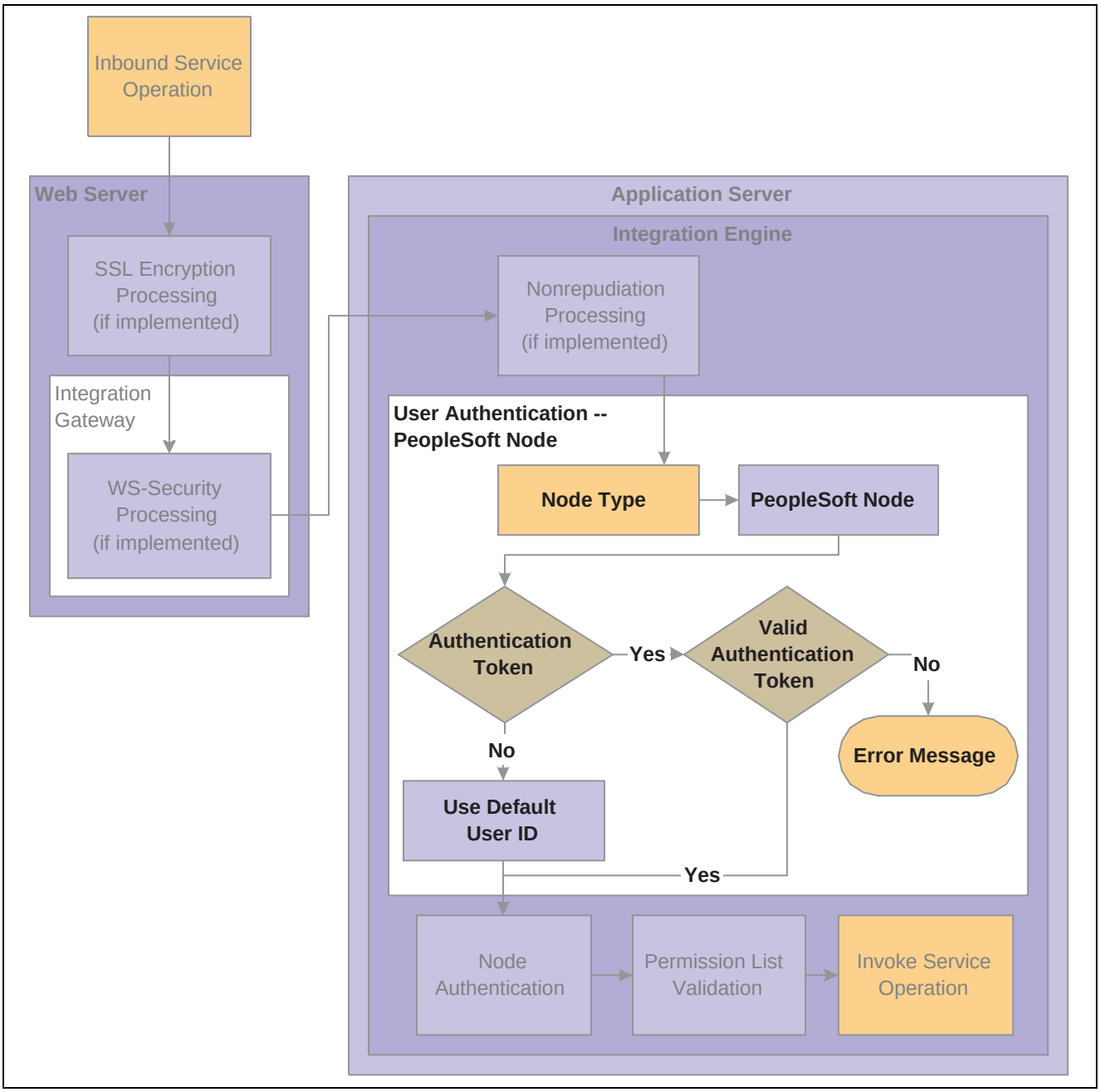
The inbound user authentication process determines the user ID that has been sent with an inbound service operation and determines if the sender is able to invoke the service operation.

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:

 Copyright © 1988-2007, Oracle. All rights reserved.



Inbound user authentication-PeopleSoft node is the sending node

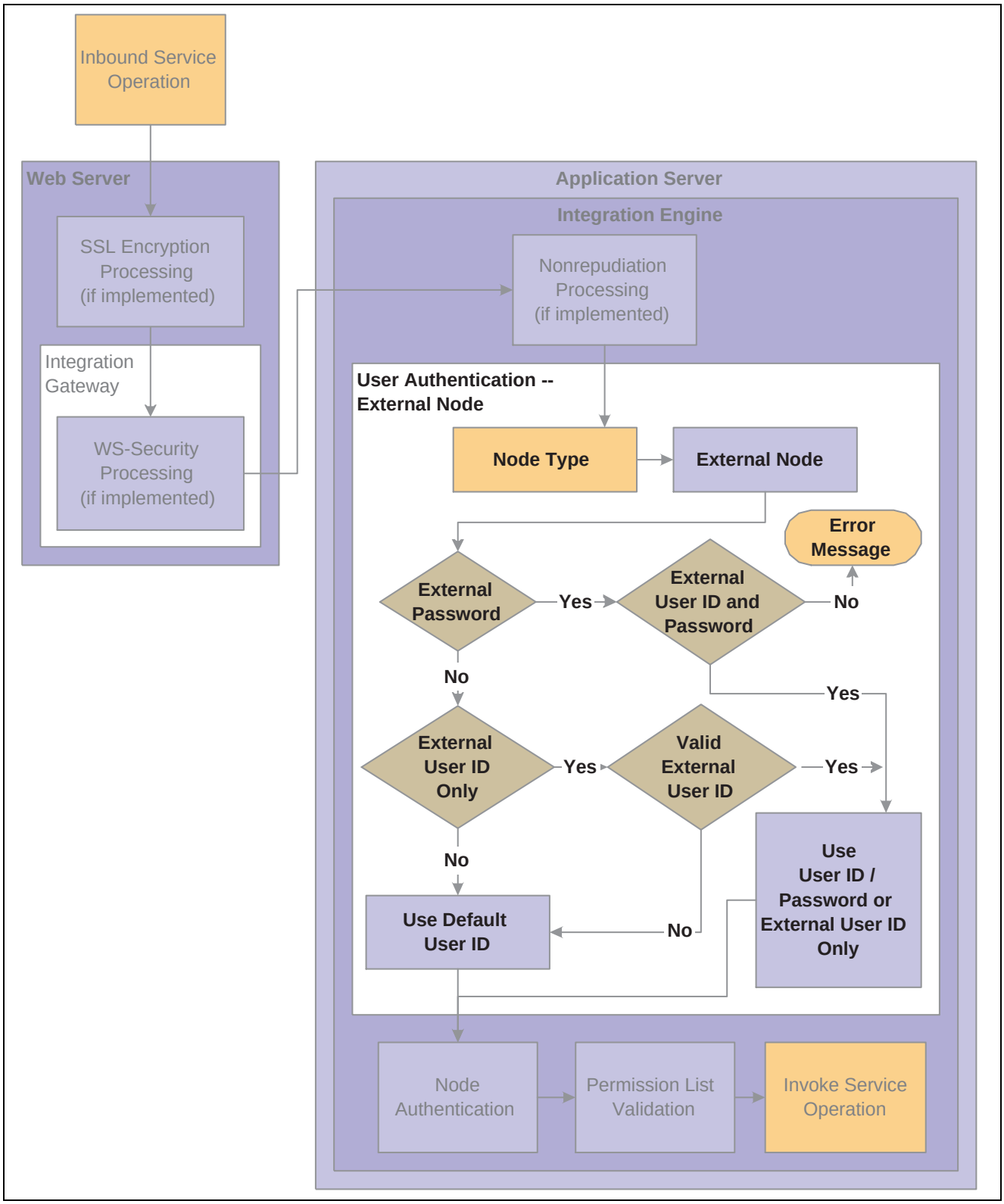
If the sending node is a PeopleSoft node, the system determines if an authentication token has been sent with the transaction. The system uses the authentication token to verify that the sending node is a trusted PeopleSoft node. If authentication passes, the service operation has passed user authentication. 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:

Copyright © 1988-2007, Oracle. All rights reserved.



Inbound user authentication-External node is the sending node

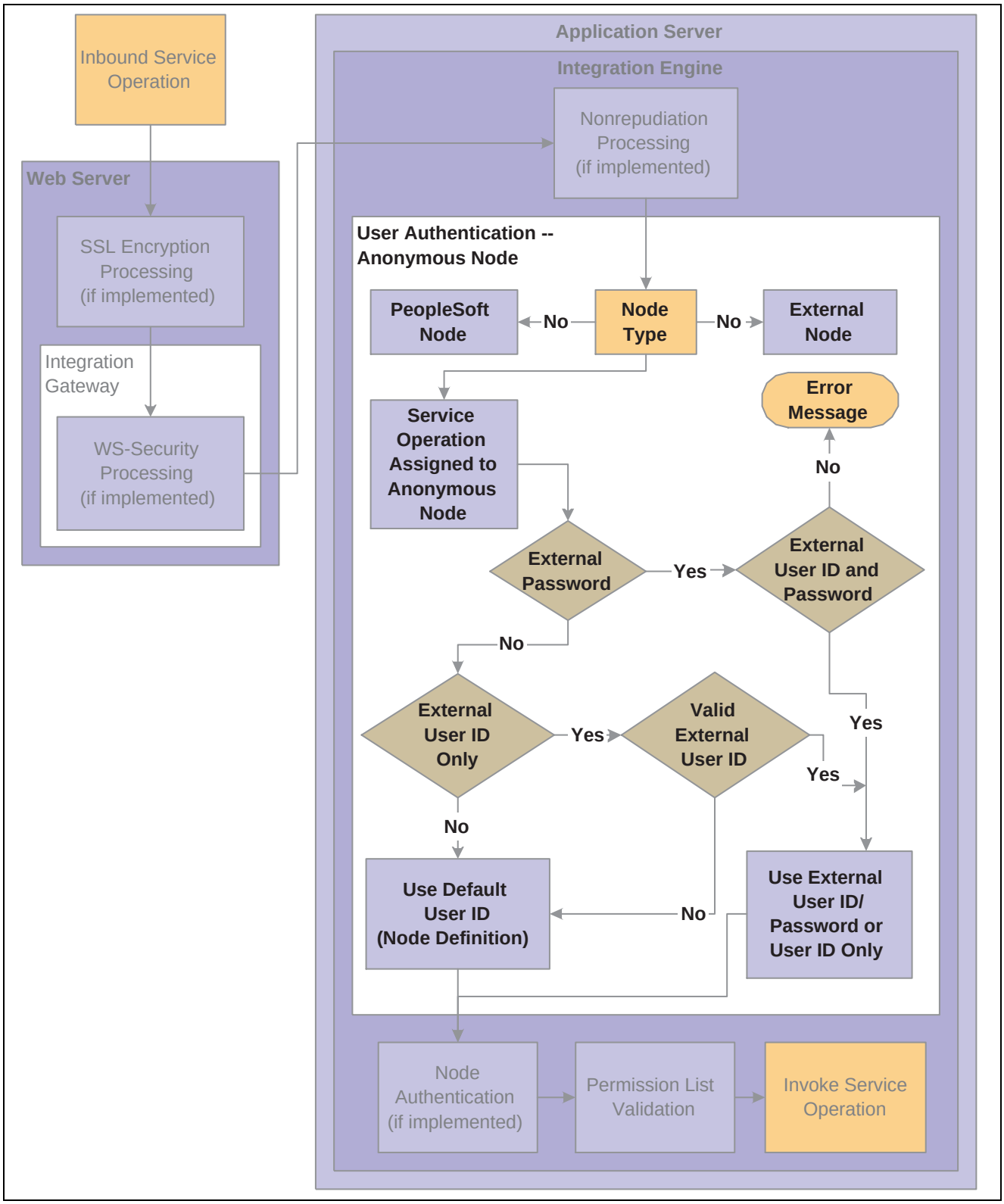
Copyright © 1988-2007, Oracle. All rights reserved.

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:

Copyright © 1988-2007, Oracle. All rights reserved.



Inbound user authentication-Third-party system sending the service operation

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.

Copyright © 1988-2007, Oracle. All rights reserved.

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.

Pages Used to Manage User Authentication

Page Name	Object Name	Navigation	Usage
Service Operations–General page	IB_SERVICE	Select PeopleTools, Integration Broker, Integration Setup, Service Operations. Click the General tab.	Activate user authentication on a service operation.
Node Definitions page	IB_NODE	Select PeopleTools, Integration Broker, Integration Setup, Nodes.	Set the user ID on a remote node. 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. You can set the user ID using the following fields on the Node Definitions page: • External User ID and External Password. • Default User ID.

Activating User Authentication on Service Operations

To activate user authentication on a service operation:

1. Access the Service Operations-General page.
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.

Copyright © 1988-2007, Oracle. All rights reserved.

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.

Setting Up User Authentication for Third-Party System

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 19, "Adding and Configuring Nodes," Defining Node Parameters, page 363

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 19, "Adding and Configuring Nodes," Defining Node Parameters, page 363.

Setting Up Certificate-Based Node Authentication

Certificate-based node authentication involves the following tasks:

 Copyright © 1988-2007, Oracle. All rights reserved.

- 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 27, "Setting Up Secure Integration Environments," Installing Application Server-Based Digital Certificates, page 591.

- 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 19, "Adding and Configuring Nodes," Defining Node Parameters, page 363.

Securing Service Operations with Permission Lists

Securing service operations with permission lists is discussed elsewhere in this PeopleBook.

See Chapter 15, "Managing Service Operations," Setting Permissions to Service Operations, page 304.

Copyright © 1988-2007, Oracle. All rights reserved.

 Copyright © 1988-2007, Oracle. All rights reserved.

CHAPTER 28

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 master-slave dispatchers.
- Configure integration gateways for load balancing.
- Implement load balance on service operation queues.
- Resubmit failed transactions.
- Utilize data mover scripts for large service operation notifications.

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

Chapter 21, "Using the Service Operations Monitor," Viewing System Performance Statistics, page 457

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:

Copyright © 1988-2007, Oracle. All rights reserved.

- 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 6, "Administering Messaging Servers for Asynchronous Messaging," Specifying Dispatcher Parameters, page 48.

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 28, "Tuning Messaging System Performance," Exception Handling for Synchronous Message Processing, page 654

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 following parameter in PSADMIN: Thread Pool Size.

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*.

 Copyright © 1988-2007, Oracle. All rights reserved.

Setting the Thread Pool Size for Asynchronous Messaging

For asynchronous messaging, set this parameter in the Settings for Publication Contract Handler section in PSADMIN.

For synchronous 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 = %InBroker.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, 0);
&return_mesages = CreateArrayRept(&MSG, 0);

&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
```

Copyright © 1988-2007, Oracle. All rights reserved.

```
&FLIGHTPLAN_RETURN = &return_mesages [2].GetRowset();
&temp = &return_mesages [2].GenXMLString();
```

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 Chapter 18, "Managing Routing Definitions," page 331.

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, 0);
&return_mesages = CreateArrayRept(&MSG, 0);

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));
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
   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;
```

Copyright © 1988-2007, Oracle. All rights reserved.

```
   &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

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 and describes how to configure pub/sub dispatchers as master or slaves.

Understanding Implementing Master-Slave Dispatchers

This section provides an overview of implementing master-slave dispatchers

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.

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 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.

 Copyright © 1988-2007, Oracle. All rights reserved.

You configure static slave domains in PSADMIN.

Failover and Master-Slave Dispatchers

You can create a slave domain for use in domain failover.

The domain with the highest priority 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 21, "Using the Service Operations Monitor," Setting Up Domain Failover, page 480.

Configuring Dynamic Slave Dispatchers

Use the Failover Configuration page in the Service Operations Monitor to configure dynamic slave dispatchers.

See Chapter 21, "Using the Service Operations Monitor," Setting Up Domain Failover, page 480.

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 8.49 PeopleBook: System and Server Administration, "Using the PSADMIN Utility"

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.

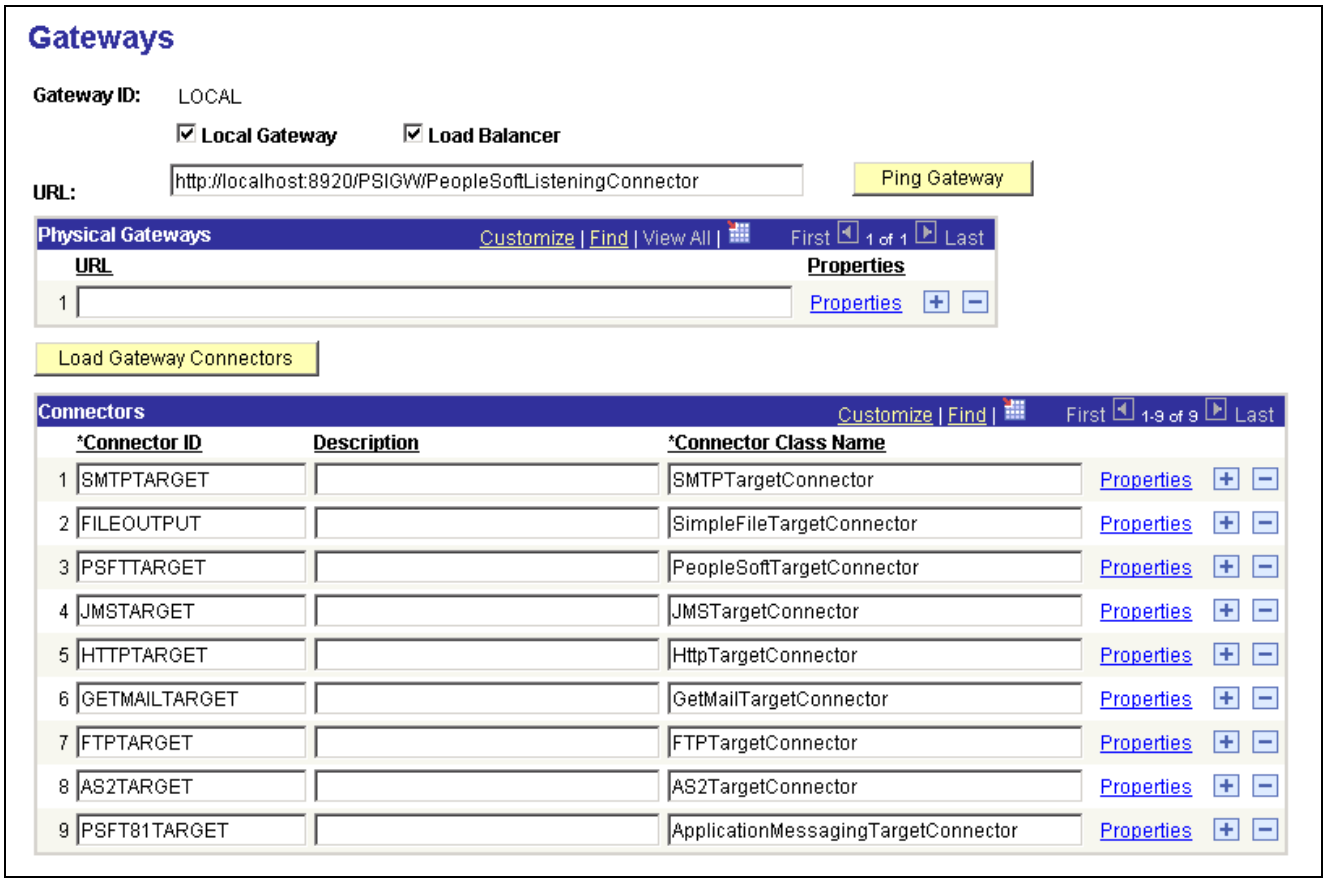
For installation and configuration information about your load balancing software, please see the documentation that is included with the product.

Copyright © 1988-2007, Oracle. All rights reserved.

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 7, “Managing Integration Gateways,” page 53

 Copyright © 1988-2007, Oracle. All rights reserved.

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.

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.

See Also

Chapter 6, "Administering Messaging Servers for Asynchronous Messaging," Specifying Dispatcher Parameters, page 48

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 8.49 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 message queues.

Copyright © 1988-2007, Oracle. All rights reserved.

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 28, "Tuning Messaging System Performance," Setting the Load Balance Interval Parameter, page 659

Chapter 6, "Administering Messaging Servers for Asynchronous Messaging," Specifying Dispatcher Parameters, page 48

Utilize Data Mover Scripts for Large Message Subscriptions

PeopleSoft Integration Broker provides a DMS handler type that serves as a bulk loader to insert data. This handler is available when working with asynchronous one-way service operation types that contain rowset-based messages.

The DMS handler performs destructive bulk inserts. This means that the data will first be deleted then the insert of the data will be performed. If the event fails the data is *not* rolled back.

See Chapter 15, "Managing Service Operations," Adding Handlers to Service Operations, page 301.

 Copyright © 1988-2007, Oracle. All rights reserved.

CHAPTER 29

Using the Inbound File Loader Utility

This chapter provides an overview of the Inbound File Loader utility processing and discusses how to:

- Set up processing rules.
- Initiate file processing.
- Test inbound flat file processing.

Understanding the Inbound File Loader Utility

When external systems send inbound transactions consisting of flat files, you can use the Inbound File Loader utility to translate the incoming files into service operations and process them.

The Inbound File Loader utility provides two options for translating and processing files:

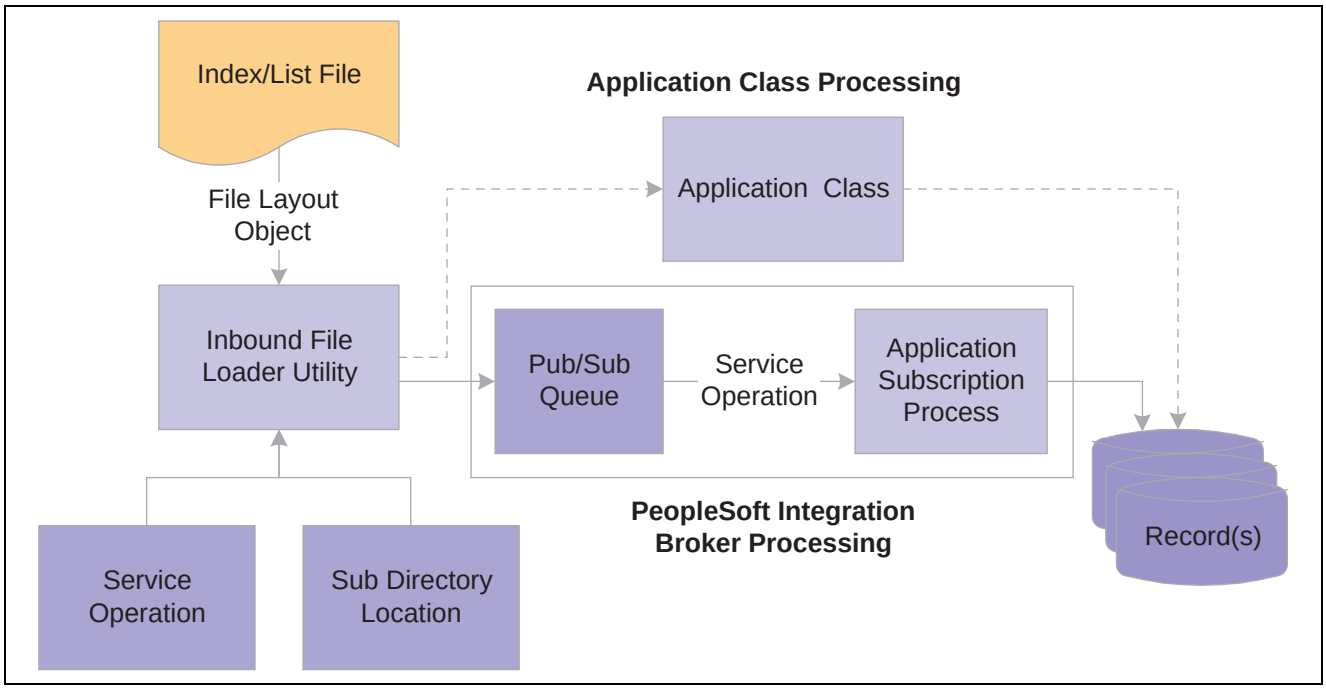
- Use PeopleSoft Integration Broker to convert file data into service operations and publish them locally. Then, subscribe to the service operations and insert the data into the tables.
- Write an application class that will read the contents of files and insert the data directly into the tables

Note. You can only use one processing method at any given time. Application class processing always overrides PeopleSoft Integration Broker processing.

File Processing

This section discusses the processing flow for the Inbound File Loader utility.

Copyright © 1988-2007, Oracle. All rights reserved.



Inbound File Loader utility processing flow

The flow for inbound file processing using the Inbound File Loader utility is:

1. The utility receives a flat file in the form of a file layout object from an external system.

   The flat file consists of either:

   - A data file that contains the relevant data.
   - An index file that contains pointers to the data.

     Each index file lists the names of a set of data files to be processed. Each line of the index file must be a plain text file that contains only one field: a file name with full directory path information.

   These files contain the application data, which is in one of the following formats: fixed record, Comma Separated Values (CSV), or XML.

   Note. The wildcards "*" and "?" may only be used in the filename and not in the directory path.

2. The utility reads the file that is submitted for processing:

   - If the file is an index file, the Inbound File Loader utility loads the list of data files that are associated with each index file to be processed into a parameter table.
   - If it is a single data file, the utility inserts the single data file into a parameter table.

     Note. If additional fields in the file layout are not in the message definition, the additional fields are ignored during the copying of the flat file data to the message and are not included in the message.

3. The utility loops through the list of data files to be processed and reads each data file.
4. The utility uses either PeopleSoft Integration Broker or application class processing logic to read the file contents and process the data.

Copyright © 1988-2007, Oracle. All rights reserved.

- PeopleSoft Integration Broker processing.

  When using Integration Broker processing logic, the Inbound File Loader utility copies the rowsets of the data files into the message, publishes the service operation locally, and then the receiving system receives the service operation and initiates normal inbound data processing.

- Application class processing.

  You can build an application class to read the contents of the inbound file as rowsets and implement the necessary processing logic to write to the underlying tables.

5. To add file archiving, delete logic to prevent files from processing again, and so on, when defining processing rules, you can optionally specify an application engine program name and section. If specified, the utility calls the program and section as a final step to the inbound file process.

Understanding Development Activities

This section discusses development activities for using the Inbound File Loader utility and describes:

- General development activities
- Development activities for PeopleSoft Integration Broker processing.
- Creating file layout definitions.
- Development activities for application class processing.

General Development Activities

This section discusses general development activities for using the Inbound File Loader utility using either PeopleSoft Integration Broker processing or application class processing.

Determining the Format for Inbound Data

Determine the necessary format of the inbound data. If there is an industry standard, use it for your file definition. If there is no industry standard, create a file layout object.

Development Activities for PeopleSoft Integration Broker Processing

This section discusses development activities for using the PeopleSoft Integration Broker processing in conjunction with the Inbound File Loader utility.

1. Create a message definition in the PeopleSoft Pure Internet Architecture.

   The structure of the message definition must be similar to the structure of the file layout definition that you created.

2. Create a service.
3. Create an asynchronous one-way service operation for the service.
4. Create a local-to-local routing definition for the service operation.
5. Create an OnNotification handler and call the functional library IB_FILE_SUBCODE.

   PeopleTools delivers this functional library and it can be used for any generic notification.

Copyright © 1988-2007, Oracle. All rights reserved.

The following example shows code an OnNotification handler calling the functional library:

```
import PS_PT:Integration:INotificationHandler;

class QUSubscribe implements PS_PT:Integration:INotificationHandler
   method QUSubscribe();
   method OnNotify(&_MSG As Message);
end-class;

Declare Function Subscribe PeopleCode FUNCLIB_IBFILE.IB_FILE_SUBCODE FieldFormula;

/* constructor */
method QUSubscribe
end-method;

method OnNotify
   /+ &_MSG as Message +/
   /+ Extends/implements PS_PT:Integration:INotificationHandler.OnNotify +/
   /* Variable Declaration */

   Local Message &MSG;
   Local Rowset &MSG_ROWSET;

   &MSG = &_MSG;

   Subscribe(&MSG);

end-method;
```

6. Define processing rules in the Inbound File Loader Rules page in the PeopleSoft Pure Internet Architecture.
7. Initiate flat file processing using the Inbound File Processing page.
8. Test the inbound flat file processing.

Creating File Layout Definitions

When you use the Inbound File Loader utility, you use file layout definitions to read and write data from flat files. Use the following guidelines when creating file layout definitions:

- Create a file layout definition with the same structure as the message definition to support the vendor file format.
- The hierarchical structure of the data in the file layout definition must match that of the message definition. For example, suppose a message has three levels:
  - Level 0, containing record A.
  - Level 1, containing records B and C.
  - Level 2, containing record D.

  All file layouts that are associated with this message must also have record A in level 0, record B and C in level 1, and record D in level 2.

 Copyright © 1988-2007, Oracle. All rights reserved.

Note. The file layout does not need to contain the exact same fields as the message definition

- For every record in a file layout definition, add a new file field, AUDIT_ACTN, as the first field in the record (except when the field already exists in the application table).
- You can associate more than one file layout to a single message. For example, vendor A may have a different number of fields than vendor B, so you may have two file layouts: one for vendor A and one for vendor B.
- Specify the file ID uniquely to include a row in a file, which is necessary in mapping the data to its proper record. Include start and end points when dealing with more than one record in a file layout.
- Each record in the file layout has a file record ID attribute. Do not confuse this with the file layout ID. The file layout ID determines whether a new layout is encountered for multiple file layout processing.

Development Activities for Application Class Processing

This section discusses development activities for application class processing using the Inbound File Loader utility. This section discusses how to:

1. Create an application class.
2. Specify processing rules.

Creating Application Classes

This section discusses creating an application class for processing flat files in conjunction with the Inbound File Loader utility.

The application class you create must implement the IProcessFile interface. The signature of the interface is shown here:

```
interface IProcessFile
   /* Any initialization is done in this function*/
method Init (&fldDefName As string, &msgName As string) Returns boolean;

/* Contains processing logic that stores the Rowset data into the respective⇒
 tables */
method Process(&fldDefName As string, &msgName As string, &rs As Rowset) Returns⇒
 boolean;

/* This method shall contain logic for cleanup operations */
method Finish(&fldDefName As string, &msgName As string) Returns boolean;
end-interface;
```

The application class you create must implement the following three methods:

- Init
- Process
- Finish

If the Replace Data check box is selected in the Inbound File Loader Rule page, the Init method will be called. The Finish method is the last method to be invoked by the utility. Any post-processing clean up code can be implemented with this function.

Copyright © 1988-2007, Oracle. All rights reserved.

The logic in the Process method stores the file contents in staging tables. You can add logic in the Finish method to move the data from staging tables to the actual transaction tables as a final process.

The Init, Process and Finish methods must return a boolean value of *True* for successful completion of the file processing. If methods Init and Finish are not used, return a default value of *True*.

The following example shows an application class implementing the IProcessFile interface:

```
import PTIB:Interfaces:IProcessFile;

class InboundFileProcess implements PTIB:Interfaces:IProcessFile

method Init(&fldDefName As string, &msgName As string) Returns boolean;

method Process(&fldDefName As string, &msgName As string, &rs As Rowset) Returns⇒
 boolean;

method Finish(&fldDefName As string, &msgName As string) Returns boolean;

end-class;

method Init
   /+ &fldDefName as String, +/
   /+ &msgName as String +/
   /+ Returns Boolean +/
/+ Extends/implements PTIB:Interfaces:IProcessFile.Init +/

//This function will be called when the Replace Data flag is
//enabled
//add initialization code, such as cleaning up the table before
//reading in the data from the file



   Return True;
end-method;

method Process
   /+ &fldDefName as String, +/
   /+ &msgName as String, +/
   /+ &rs as Rowset +/
   /+ Returns Boolean +/
/+ Extends/implements PTIB:Interfaces:IProcessFile.Process +/

//Add the code that inserts/updates/delete data in the table

   Return True;
end-method;

method Finish
   /+ &fldDefName as String, +/
   /+ &msgName as String +/
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
   /+ Returns Boolean +/
   /+ Extends/implements PTIB:Interfaces:IProcessFile.Finish +/

 //This function will be called when the Replace Data flag is
//enabled
// Clean up logic goes here (if any)

   Return True;
end-method;
```

Specifying Processing Rules

After you create an application class, you must access the Inbound File Loader Rules page and specify the following information:

- Root Package ID.
- Path.
- Class name.

Prerequisites for Using the Inbound File Loader Utility

The prerequisites for using the Inbound File Loader utility are:

- PeopleSoft Integration Broker must be configured and running.
- PeopleSoft Process Scheduler must be configured in PSADMIN.
- Create a file definition layout as described previously in this chapter.

  See Chapter 29, "Using the Inbound File Loader Utility," Creating File Layout Definitions, page 664.
- If using PeopleSoft Integration Broker processing, complete the development activities for PeopleSoft Integration Broker processing described previously in this chapter.

  See Chapter 29, "Using the Inbound File Loader Utility," Development Activities for PeopleSoft Integration Broker Processing, page 663.
- If using application class processing develop an application class that implements the IProcessFile interface and specify the processing rules as described previously in this chapter.

  See Chapter 29, "Using the Inbound File Loader Utility," Development Activities for Application Class Processing, page 665.

Setting Up Inbound File Loader Processing Rules

This section discusses how to set up inbound flat file processing rules.

Copyright © 1988-2007, Oracle. All rights reserved.

Understanding Setting Up Inbound File Loader Processing Rules

The Inbound File Loader utility uses information you define in the Inbound File Loader Rules page to determine the file layout and message combination, as well as other file attributes necessary for processing files.

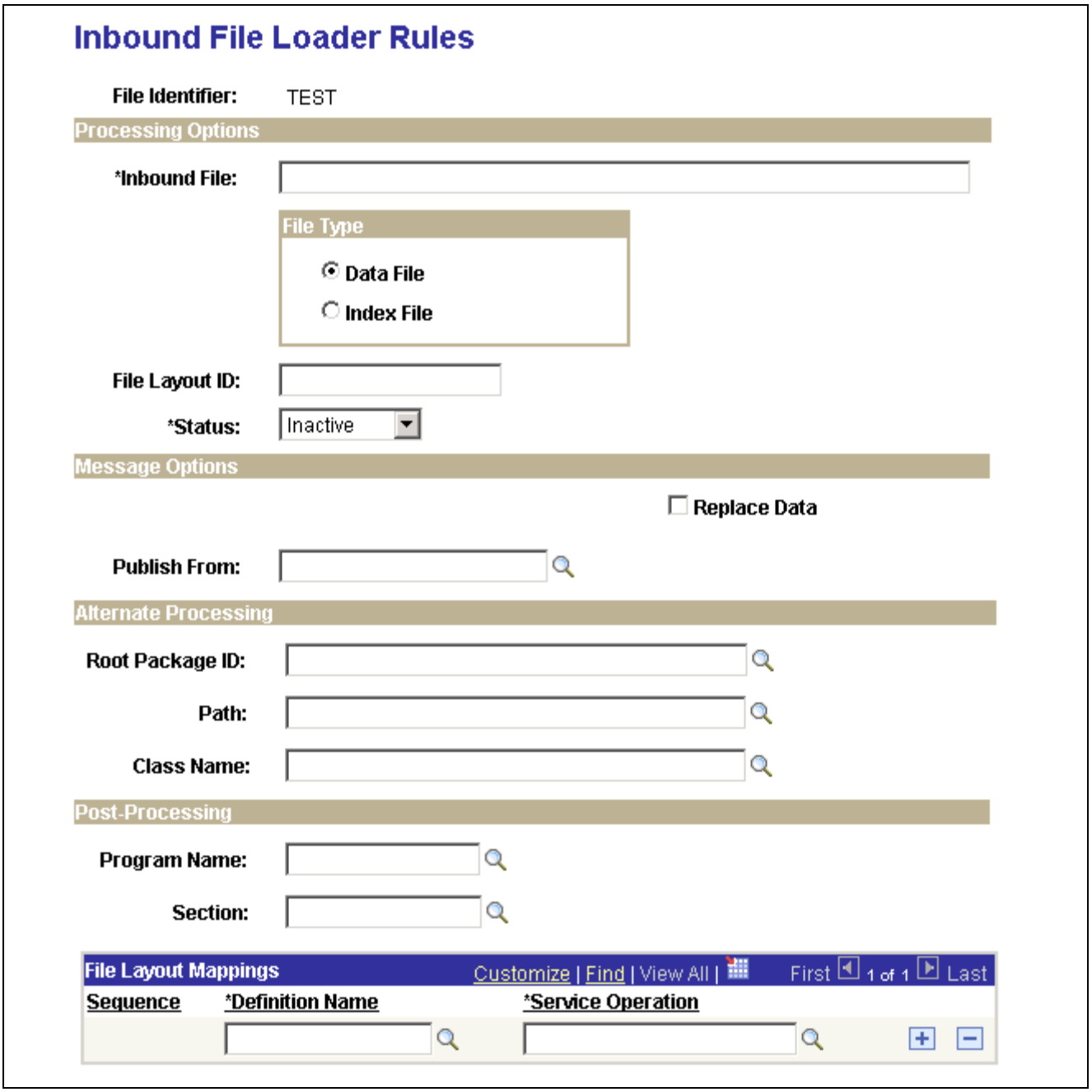
Page Used to Set Up Inbound File Loader Processing Rules

Page Name	Object Name	Navigation	Usage
Inbound File Loader Rules page	PSIBINFILERULE	Select PeopleTools, Integration Broker, File Utilities, Inbound File Loader Rules.	Use the page to specify the file layout and message to process, as well as parameters for processing.

Setting Up Inbound File Loader Processing Rules

Use the Inbound File Loader Rules page to specify the file layout and message to process, as well as define the parameters for processing. To access the page, select PeopleTools, Integration Broker, File Utilities, Inbound File Loader Rules. The following examples shows the page:

Copyright © 1988-2007, Oracle. All rights reserved.



Inbound File Loader Rules page

Note. You can process multiple inbound flat files at one time. By specifying an inbound index file as part of the Inbound File Loader utility parameters. The system reads all input files within the index file and uses the associated file layout object and message to convert the data. Similarly, specify a wildcard in the filename in the inbound file rule component, but make sure that all files that meet the wildcard criteria correspond to the file layout and message mapping that are defined.

File Identifier Displays the inbound file that you are associating with the rule.

Inbound File Enter the index file name or the data file name of the inbound file to process. Specify the full path information. The PeopleCode program uses the *%filepath_absolute* variable when opening the file.

Copyright © 1988-2007, Oracle. All rights reserved.

File Type In the File Type section, select the type of inbound file. The options are:

- Data File
- Index File

File Layout ID (Optional.) Enter a file layout ID to associate with the file. The file layout ID is used in a multiple file layout processing. This identifier indicates that the data that follows belongs in a new file layout.

Status From the Status dropdown list, select whether this inbound file rule is active. The valid options are:

- Active.
- Inactive. (Default.)

Replace Data Check the box to indicate that the inbound file processing is a destructive load process.

A destructive load process involves replacing the contents of the tables with new data from the file being processed. In a destructive load process, the service operations must be subscribed in the same order as they are published to ensure transactional integrity.

If this check box is selected, the utility publishes a header message. The header message is a trigger in the subscription process to initialize tables before receiving the data messages. The subscription PeopleCode logic must check for the header message and perform any cleanup operation .

The utility then publishes data messages containing the data followed by a Trailer Message. The trailer message is used as a trigger in the subscription process to indicate that all the data messages have been received.

If the Replace Data check box is not selected, only data messages are published.

When used in the context of application class processing, if the Replace Data flag is selected, the Init() and Finish() methods of the specified application class are invoked.

Publish From Select the name of the publishing node.

Use this option to simulate an inbound asynchronous transaction from an external node.

While using this feature, you must create an inbound asynchronous transaction on the node from where the message is published.

Root Package ID Select the root package ID of the application class.

Path Select the qualifying path of the application class

Class Name Select the application class name that contains the file processing logic

Program Name and**Section** Select a PeopleSoft Application Engine program and section to invoke when the utility finishes processing data.

Definition Name Specify the file layout definition for the file(s) being processed.

If the File Layout ID field is blank, this field should contain only one entry

If the File Layout ID field is not blank, this scroll area must contain an entry for each file layout definition name that is specified in the inbound file.

 Copyright © 1988-2007, Oracle. All rights reserved.

Service Operation For every row in this scroll area, specify a service operation. The utility uses the message(s) defined within the service operation to copy the file rowsets into message rowsets

Note. Use the wildcards * and ? for the file name but not for the directory path. The file layout and service operation must be valid for all files that meet the wildcard criteria.

Initiating File Processing

This section discusses how to initiate inbound flat file processing.

Understanding Initiating File Processing

The Inbound File Processing page runs the Application Engine process PTIB_INFILE that initiates the file-to-message processing. The file-to-message processing function reads the file rowset and publishes it as a message.

If an index file exists when the inbound conversion process runs, the application engine program loads the list of files to be converted into a parameter table and completes a commit. The application engine program uses the list of files within the parameter table to restart the processing if a particular flat file fails. If a single data file is provided, then the rowset processing immediately begins.

The file publish process goes through each of the rowsets of the file layout and copies them into the message row sets.

If the audit action (AUDIT_ACTN) exists in the file, it is copied to the PSCAMA record. If the audit action does not exist in the file, the publishing process uses the default value that is specified in the file layout field property.

The Inbound File Loader utility publishes a new message when one of the following exists:

- The size of the data in the service operation exceeds the value of the Maximum App Message Size field set in the PeopleTools Options page.

  To view the value in the field, select PeopleTools, Utilities, Administration, PeopleTools Options.
- A new file layout is detected.
- End of file is reached.

The application engine program completes a commit every time a message is published from a file. After conversion, the flat file remains in the parameter table with a status of *Processed.*

Note. The file layout should exactly match the message layout (excluding the PSCAMA record) and should use the same character set as that used by the file: either American National Standards Institute or Unicode.

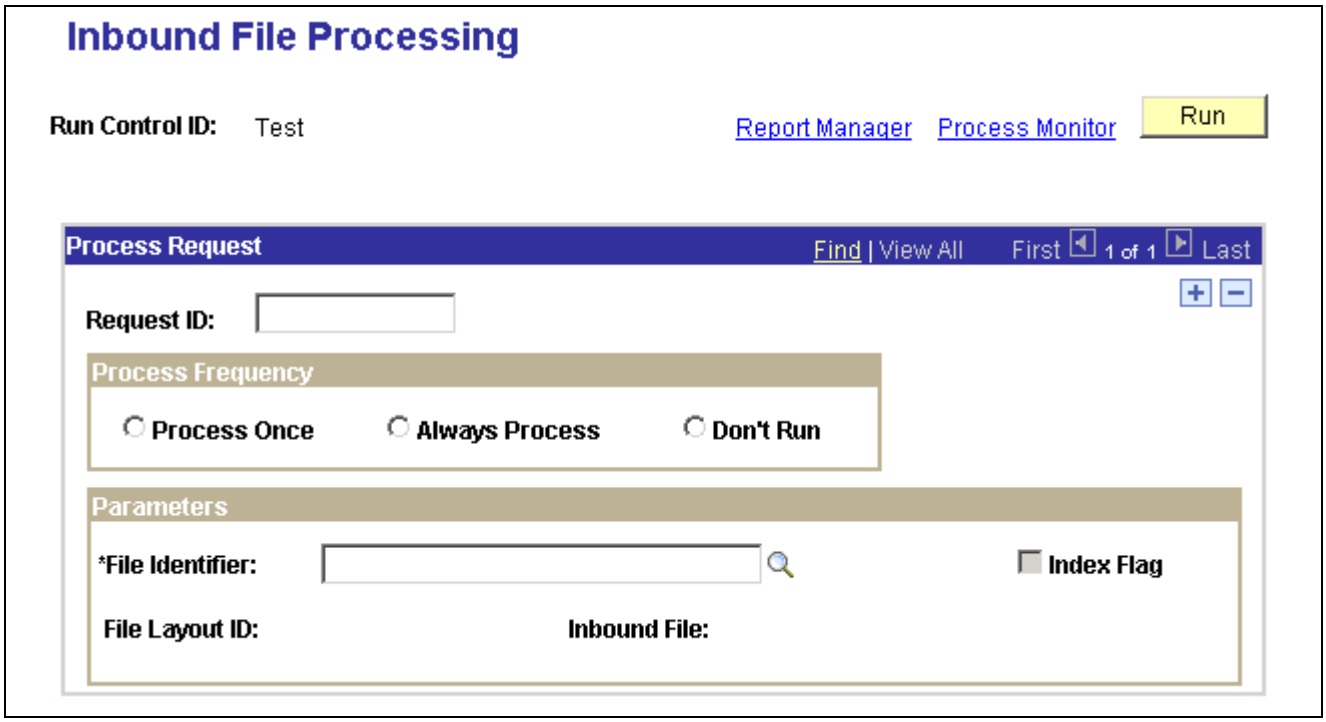
Page Used to Initiate Inbound Flat File Processing

Page Name	Object Name	Navigation	Usage
Inbound File Processing page	PSIBFILERUNCNTL	Select PeopleTools, Integration Broker, File Utilities, Inbound File Processing.	Use the page to initiate inbound flat file processing.

Copyright © 1988-2007, Oracle. All rights reserved.

Initiating Inbound Flat File Processing

Use the Inbound File Processing page to initiate flat file processing. Select PeopleTools, Integration Broker, File Utilities, Inbound File Processing. The Inbound File Processing page, shown in the following example, appears:



Inbound File Processing page

Request ID Enter a unique identifier to track the request.

Process Frequency Select the frequency for processing the file. The options are:

- Process Once.
- Always.
- Don't Run.

File Identifier Select or enter the name of the file identifier that you set up in the Inbound File Loader Rules. page. The file identifier is tied to the publish rules.

Index Flag A read-only field that indicates if the file being processed is an index file or a data file. When checked, the Inbound File Loader utility is processing an index file.

File Layout ID A read-only field that displays the file layout ID associated with the file in the Inbound File Loader Rules page.

Inbound File A read-only field that displays the name of the file being processed

Copyright © 1988-2007, Oracle. All rights reserved.

Testing Inbound Flat File Processing

To test inbound files:

1. Create a sample flat file, or ask the third-party vendor for a sample flat file.
2. Set up processing rules for the test.

   Select PeopleTools, Integration Broker, File Utilities, Inbound File Loader Rules to access the Inbound File Loader Rules page.

   See Chapter 29, "Using the Inbound File Loader Utility," Setting Up Inbound File Loader Processing Rules, page 668.
3. Initiate file processing.

   Select PeopleTools, Integration Broker, File Utilities, Inbound File Processing to access the Inbound File Processing page.

   See Chapter 29, "Using the Inbound File Loader Utility," Initiating Inbound Flat File Processing, page 672.
4. Verify the inbound processing created a message that contains the sample flat file data.
   a. If you used application class processing, verify if the production or staging tables are loaded with the correct field values. For production tables, look in the PeopleSoft application pages. For staging tables, use either the PeopleSoft application pages or run a query by using PeopleSoft Query.
   b. If you used PeopleSoft Integration Broker for file processing, use the Service Operations Monitor to ensure the inbound file processing created a service operation that contains the sample flat file data.

      Verify that the standard inbound notification process received the message and processed it into the application tables.

      Determine whether the values become the inherited values (if you used the inherited value feature in file layout).

      Validate that the production or staging tables loaded with the correct field values. For production tables, look in the PeopleSoft application pages. For staging tables, use either the PeopleSoft application pages or run a query by using PeopleSoft Query.

      Ensure that the date formats conform.

Copyright © 1988-2007, Oracle. All rights reserved.

 Copyright © 1988-2007, Oracle. All rights reserved.

CHAPTER 30

Backporting Integration Metadata

This chapter provides an overview of backporting integration metadata and discusses how to:

- Use the Metadata Backport utility.
- Work with backported projects.
- Clean up PeopleTools 8.49 databases after backporting metadata.

Understanding Backporting Integration Metadata

The Backport Metadata utility clones a PeopleTools 8.49 PeopleSoft Application Designer project that you specify to a project that you can use on PeopleTools 8.4x systems running releases prior to PeopleTools 8.48.

Metadata Backported

This section discusses the integration data that the Metadata Backport utility backports.

Message Queues

The Metadata Backport utility backports PeopleTools 8.49 queues to channels that were used in PeopleTools 8.4x releases prior to PeopleTools 8.48. The queue name becomes the channel name in the cloned project.

Integration PeopleCode

The Metadata Backport utility also backports PeopleTools 8.49 application class handlers to integration PeopleCode constructs that were used in previous PeopleTools 8.4x releases prior to PeopleTools 8.48.

Only application classes that are used as part of a handler are included in the backport.

The following table lists how the utility backports handlers based on the following application classes and application class methods:

Integration Broker 8.49 Application Class/Method Name	Backported PeopleCode
INotificationHandler/OnNotify.	Subscription event.
IReceiveHandler/OnAckReceive.	OnAckReceive event.
IRequestHandler//OnRequest.	OnRequest event.

Copyright © 1988-2007, Oracle. All rights reserved.

Integration Broker 8.49 Application Class/Method Name	Backported PeopleCode
IRouter/OnRouteReceive or OnRouteSend.	OnRouteReceive event or OnRouteSend event.
ISend/OnRequestSend.	OnSend event.

In the case for backporting handlers implementing the iNotificationHandler interface, the intent is to name the subscription to the same name as the PeopleTools 8.49 application class. However, due do their more restrictive naming convention—for example, no special characters, no spaces, and so on—instances may occur in which the backported message subscription name does not match the original application class name.

Using the Metadata Backport Utility

This section discusses using the Metadata Backport utility to backport integration metadata.

Page Used to Backport Integration Metadata

Page Name	Object Name	Navigation	Usage
Metadata Backport page	PTIB_BP_RUNCNTL	Select PeopleTools, Integration Broker, Service Utilities, Metadata Backport	Backport PeopleTools 8.49 integration metadata to a project that you can use in PeopleTools 8.4x releases prior to PeopleTools 8.48.

Backporting Integration Metadata

Access the Metadata Backport page. The following example shows the page.

Metadata Backport page

To backport integration metadata:

1. In the Project Name field on the Metadata Backport page, enter the PeopleTools 8.48 Application Designer project to backport.
2. In the Output Project Name field, enter the name for the backported project.
3. Click the Run Now button.

   A Proceed with backport conversion? dialog box appears.

Copyright © 1988-2007, Oracle. All rights reserved.

4. Click the Yes button.

Working with Backported Projects

After running the Metadata Backport utility, use the PeopleSoft Application Designer Copy to File feature to copy the backported project to file. Use the PeopleSoft Application Designer Copy From File feature to copy the file to the appropriate pre-PeopleTools 8.49 database.

See Also

Enterprise PeopleTools 8.49 PeopleBook: PeopleSoft Application Designer, "Copying Projects and Definitions," Copying Projects

Cleaning Up PeopleTools 8.49 Databases After Backporting Metadata

During the backport process, the Metadata Backport utility creates channels, application classes and integration PeopleCode on the PeopleTools 8.49 database. The Metadata Backport utility creates a DELETE project you can run to remove this data from the database. The DELETE project contains similar items to your clone project but has those items marked with an upgrade action of Delete in the project. The name of the delete project the utility creates takes the format *<output_project_name>DELETE*.

After you copy the backported project from the current database to file, you can re-import the DELETE project to remove the pre-PeopleTools 8.49 integration metadata from that database. To do this, copy the DELETE project to file. Then, leverage the Copy From File feature in PeopleSoft Application Designer to copy the DELETE project back into the database, thereby removing the channels, application classes and message PeopleCode created by the backport program from the PeopleTools 8.49 database. This ensures that the deprecated objects are removed from the database.

See Also

Enterprise PeopleTools 8.49 PeopleBook: PeopleSoft Application Designer, "Copying Projects and Definitions," Copying Projects

Copyright © 1988-2007, Oracle. All rights reserved.

 Copyright © 1988-2007, Oracle. All rights reserved.

APPENDIX A

Integration Scenarios

This appendix provides an overview of the basic integration scenarios you can implement using PeopleSoft Integration Broker and discusses how to:

- Integrate with PeopleSoft Integration Broker systems.
- Integrate with PeopleSoft Integration Broker systems through a firewall.
- Integrate with PeopleSoft Integration Broker systems by using hubs.
- Integrate with third-party systems.
- Integrate with third-party systems by using remote gateways.
- Integrate with PeopleSoft 8.1x systems.

Understanding Integration Setup

An integration engine is automatically installed as part of your PeopleSoft application, and an integration gateway is installed as part of the PeopleSoft Pure Internet Architecture. However, there's no requirement that your integration gateway be on the same machine as the integration engine.

In general, the high-level tasks that you perform to configure any of the integration scenarios are:

- Define a local integration gateway.
- Define a remote integration gateway.
- Set integration gateway properties.
- Set up a local node.
- Set up a remote node.
- Create service operations with inbound and outbound routing definitions.

You may not need to perform all of these tasks. For example, if you don't need to communicate through a firewall, you probably don't need to define a remote gateway.

Application messaging, the precursor to PeopleSoft Integration Broker, employed content-based routing—each message had to provide its own routing information, which was defined in the message header. With PeopleSoft Integration Broker, you define the routing information separately. You can apply multiple routings to a message and change the routings independent of the message definition.

Copyright © 1988-2007, Oracle. All rights reserved.

Defining Local and Remote Integration Gateways

On each PeopleSoft Integration Broker system in your configuration, you must specify a local integration gateway. The local gateway is the application's first point of contact with other PeopleSoft applications, third-party systems, PeopleSoft Integration Broker hubs, and remote gateways. You must define exactly one local gateway for each integration engine, but a single installed gateway can serve multiple engines. The web server where the integration gateway resides can be any machine on which you've installed the PeopleSoft Pure Internet Architecture.

To define the local gateway, use the Gateways component (IB_GATEWAY) to:

- Add the gateway that the application will use to communicate with other systems.
- Specify the uniform resource locator (URL) of the gateway's PeopleSoft listening connector.

  This is the connector that receives messages from an integration engine (including the default local node) or another integration gateway.
- Register the target connectors that are delivered with PeopleSoft Integration Broker.

  These target connectors are automatically installed during the PeopleSoft Pure Internet Architecture installation process. When you subsequently define local and remote nodes, you specify—from the list of installed target connectors—which connector the local gateway should use to send messages to each node.

Note. The remote gateway default connector setting in the integrationGateway.properties file, ig.connector.defaultremoteconnector, determines through which connector the gateway routes messages that are bound for other gateways. By default, this property is set to the HTTP target connector, HTTPTARGET. Never change this setting unless you develop a custom connector to handle this routing.

An integration gateway also includes a set of listening connectors, which are likewise installed with the PeopleSoft Pure Internet Architecture. You don't to need to specify these connectors directly; third-party systems send messages to the gateway by specifying the URL of an appropriate listening connector.

Setting Integration Gateway Properties

You set integration gateway properties by using the gateway's primary configuration file, called integrationGateway.properties. You must set a BEA Jolt connect string in this file to enable communication with each PeopleSoft Integration Broker node that will be involved in an integration that uses this gateway.

The following example shows the required BEA Jolt connect string setting:

```
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>
```

Note. You can also configure a default BEA Jolt connect string to specify the target node to use when a message arrives at the gateway but doesn't specify a node name matching any of the existing entries. The default BEA Jolt connect string settings are identical to the others, except that they don't include a node name. You can specify any PeopleSoft Integration Broker node as the default node, including any of the existing entries.

Configuring Local and Remote Nodes

Use the Nodes component (IB_NODE) to configure local and remote nodes.

When you configure nodes, use the Nodes - Connectors page to specify the gateway and target connector to use to send messages to each node. Use the information in the following table as a guide for choosing the appropriate information for the configuration scenarios that are described in this appendix:

 Copyright © 1988-2007, Oracle. All rights reserved.

Scenario	System	Node Definition	Connector
PeopleSoft as a web service provider.	NA	NA	Not applicable (NA)*
PeopleSoft Integration Broker to PeopleSoft Integration Broker	Both applications	Default local	NA*
PeopleSoft Integration Broker to PeopleSoft Integration Broker	Both applications	Remote	PSFTTARGET (PeopleSoft target connector)
PeopleSoft Integration Broker to PeopleSoft systems.	Both applications	Default local	NA*
PeopleSoft Integration Broker to PeopleSoft systems.	Both applications	Remote	PSFTTARGET (PeopleSoft target connector)
PeopleSoft Integration Broker to PeopleSoft Integration Broker by using a remote gateway	Both applications	Default local	NA*
PeopleSoft Integration Broker to PeopleSoft Integration Broker by using a remote gateway	Both applications	Remote	PSFTTARGET
PeopleSoft Integration Broker to PeopleSoft Integration Broker by using a hub	Both applications	Default local	NA*
PeopleSoft Integration Broker to PeopleSoft Integration Broker by using a hub	Both applications	Remote	PSFTTARGET
PeopleSoft Integration Broker to PeopleSoft Integration Broker by using a hub	PeopleSoft Integration Broker hub	Default local	NA*
PeopleSoft Integration Broker to PeopleSoft Integration Broker by using a hub	PeopleSoft Integration Broker hub	Remote	PSFTTARGET
PeopleSoft Integration Broker to a third party	PeopleSoft Integration Broker	Default local	NA*

Copyright © 1988-2007, Oracle. All rights reserved.

Scenario	System	Node Definition	Connector
PeopleSoft Integration Broker to a third party	PeopleSoft Integration Broker	Remote	Third-party connector, as appropriate: • HTTPTARGET (HTTP target connector) • JMSTARGET (Java Message Service [JMS] target connector) target connector) • FTPTARGET (File Transfer Protocol [FTP] target connector) • SMTPTARGET (Simple Mail Transfer Protocol [SMTP] target connector)
PeopleSoft Integration Broker to a third party	Third-party system	NA	NA
PeopleSoft Integration Broker to a third party by using a remote gateway	PeopleSoft Integration Broker	Default local	NA*
PeopleSoft Integration Broker to a third party by using a remote gateway	PeopleSoft Integration Broker	Remote	Third-party connector, as appropriate: • HTTPTARGET • JMSTARGET • FTPTARGET • SMTPTARGET
PeopleSoft Integration Broker to a third party by using a remote gateway	Third-party system	NA	NA
PeopleSoft Integration Broker to PeopleSoft 8.1x	PeopleSoft Integration Broker	Default local	NA*

Copyright © 1988-2007, Oracle. All rights reserved.

Scenario	System	Node Definition	Connector
PeopleSoft Integration Broker to PeopleSoft 8.1x	PeopleSoft Integration Broker	Remote	PSFT81TARGET (PeopleSoft 8.1 target connector)
PeopleSoft Integration Broker to PeopleSoft 8.1x	PeopleSoft 8.1x system	NA Set up message nodes, message channels, messages, and so on.	NA

* The default is PSFTTARGET, but it is not used.

See Also

Chapter 7, “Managing Integration Gateways,” Using the integrationGateway.properties File, page 64

Chapter 7, “Managing Integration Gateways,” Administering Integration Gateways, page 54

Chapter 19, “Adding and Configuring Nodes,” page 361

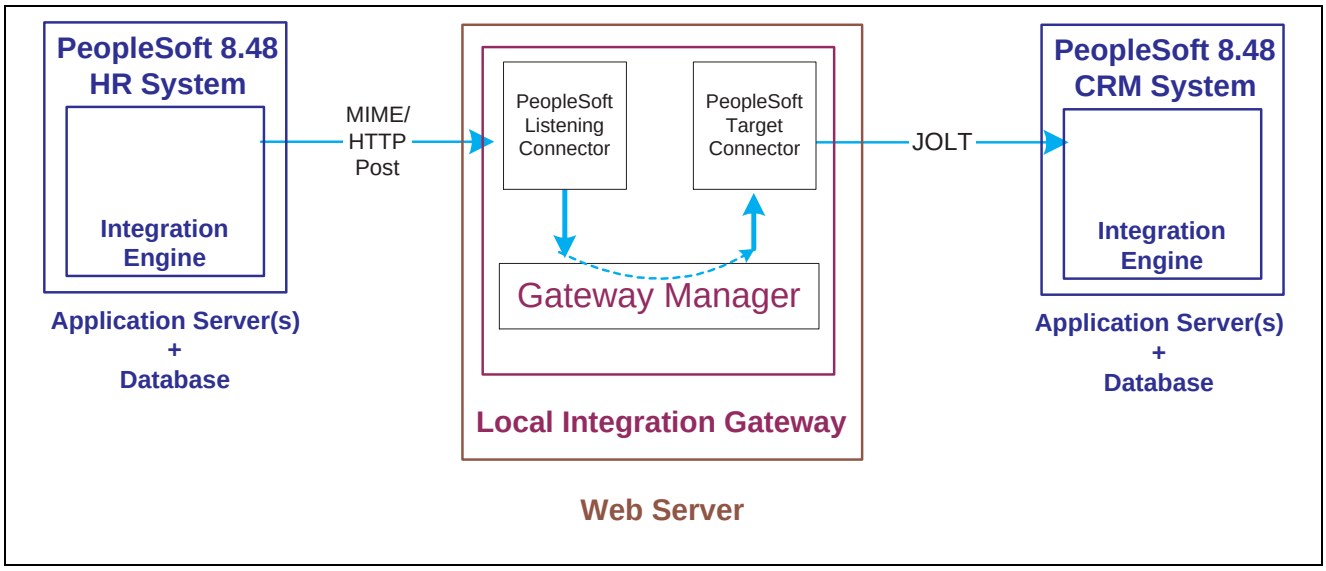
Integrating with PeopleSoft Integration Broker Systems

This section provides an overview of this scenario and discusses how to configure the system for this scenario.

Understanding This Scenario

This diagram shows a PeopleSoft Human Resources system communicating with a PeopleSoft Customer Relationship Management (PeopleSoft CRM) system and shows the configuration and interaction of PeopleSoft Integration Broker components:

Copyright © 1988-2007, Oracle. All rights reserved.



Integrations with PeopleSoft Integration Broker systems

This communication can be synchronous or asynchronous.

Configuring the System for This Scenario

This section describes the source and destination system configuration tasks, based on the scenario shown in the previous diagram. In this example, the PeopleSoft Human Resources system is the source system and the PeopleSoft CRM system is the destination system.

This section discusses how to configure:

- The integration gateway.
- The PeopleSoft Human Resources system.
- The PeopleSoft CRM system.

Configuring the Integration Gateway

The only required property that you must set for the local gateway is the BEA Jolt connect strings that enable the gateway to find the PeopleSoft CRM system. Set this property in the integrationGateway.properties file.

Configuring the PeopleSoft Human Resources System

Perform the following tasks on the PeopleSoft Human Resources system:

1. Define the local integration gateway for the PeopleSoft Human Resources system by using the Gateways component.

   Any integration gateway that you've installed and configured to find the PeopleSoft CRM system can serve this role. Specify the gateway's PeopleSoft listening connector as the gateway's URL.

2. Configure the default local node definition that represents the PeopleSoft system by using the Nodes component.

   This node is delivered predefined on the system.

3. Define a remote node to represent the PeopleSoft CRM system.

Copyright © 1988-2007, Oracle. All rights reserved.

Because the PeopleSoft CRM system uses PeopleSoft Integration Broker, specify the local gateway for the PeopleSoft Human Resources system and its PeopleSoft target connector on the Node Definitions - Connectors page.

4. Define a service operation that specifies the request message, the service operation handler definition and routing definition.

   The routing is a point-to-point routing where the PeopleSoft CRM node is the receiving node and the PeopleSoft HR system is the sending node.

Configuring the PeopleSoft CRM System

Perform the following tasks on the PeopleSoft CRM system:

1. Define the local integration gateway for the PeopleSoft CRM system by using the Gateways component.

   Any integration gateway that you've installed and configured can serve this role, including the local gateway for the PeopleSoft Human Resources system. Specify the gateway's PeopleSoft listening connector as the gateway's URL.

2. Configure the default local node definition that represents the PeopleSoft CRM system by using the Nodes component.

   This node is delivered predefined on the system.

3. Define a remote node to represent the PeopleSoft Human Resources system.

   Because the PeopleSoft Human Resources system uses PeopleSoft Integration Broker, specify the local gateway for the PeopleSoft CRM system and its PeopleSoft target connector on the Node Definitions - Connectors page.

4. Define a service operation that specifies the request message, the service operation handler definition and routing definition.

   The routing is a point-to-point routing where the PeopleSoft CRM node is the receiving node and the PeopleSoft HR system is the sending node.

5.

See Also

Appendix A, "Integration Scenarios," Understanding Integration Setup, page 679

Chapter 15, "Managing Service Operations," Adding Service Operation Definitions, page 298

Chapter 15, "Managing Service Operations," Configuring Service Operation Definitions, page 298

Chapter 15, "Managing Service Operations," Configuring Service Operation Definitions, page 298

Chapter 7, "Managing Integration Gateways," Administering Integration Gateways, page 54

Chapter 19, "Adding and Configuring Nodes," page 361

Integrating with PeopleSoft Integration Broker Systems Through Firewalls

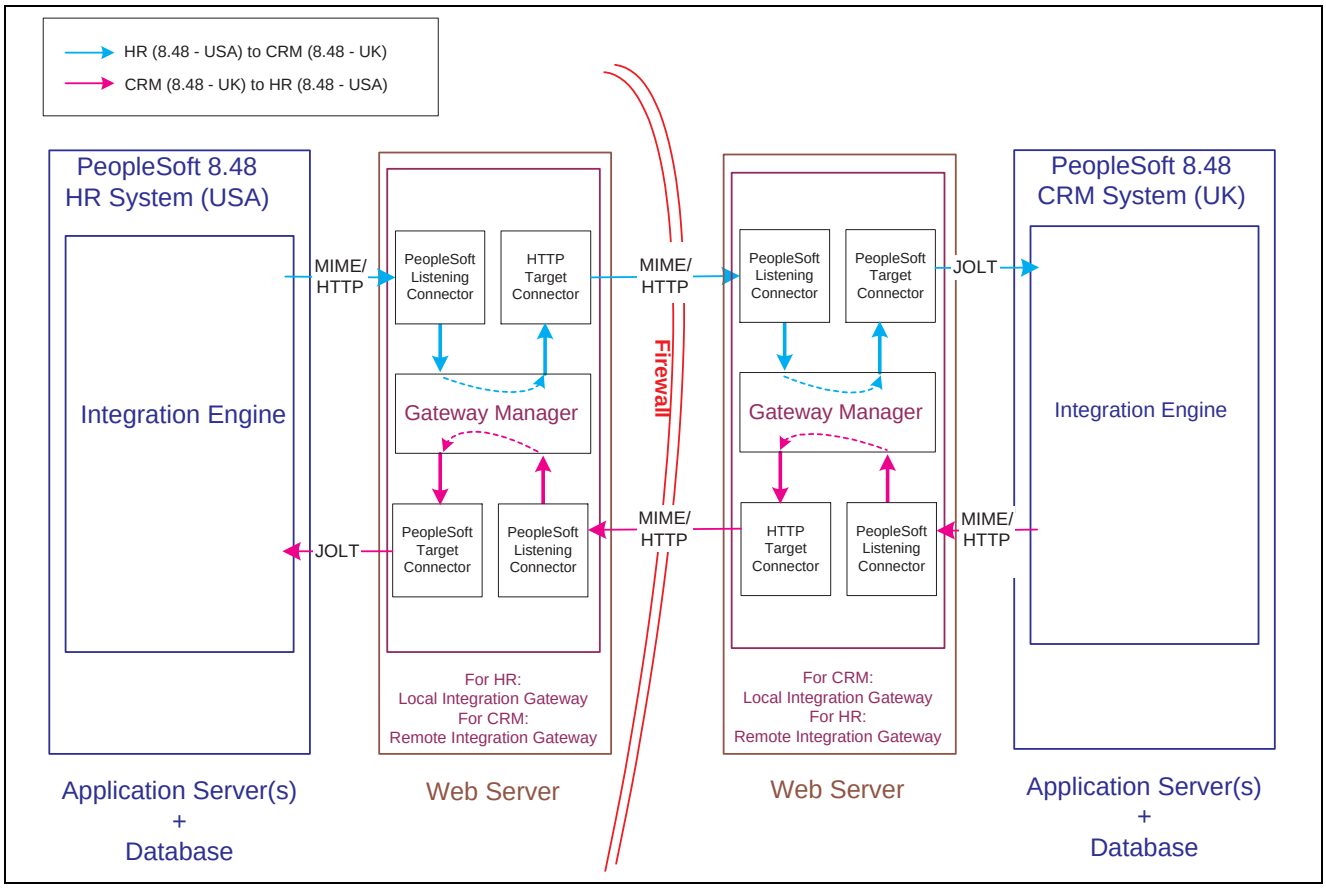
This section provides an overview of this scenario and discusses how to configure the system for this scenario.

Copyright © 1988-2007, Oracle. All rights reserved.

Understanding This Scenario

Use a remote gateway configuration when connections with an integration participant are not possible through the internet. This type of implementation enables you to communicate with wide area networks (WANs) and local area networks (LANs) where a firewall is present.

This diagram shows the configuration of PeopleSoft Integration Broker components for integrations with other PeopleSoft Integration Broker systems by using a remote gateway:



Integrations with PeopleSoft Integration Broker systems by using remote gateways

For this configuration scenario, one PeopleSoft application and one integration gateway reside on each side of the firewall. The integration gateway can reside on the same physical machine on which you have installed the PeopleSoft application, or it can reside on its own machine.

In this configuration scenario, PeopleSoft Integration Broker uses the default remote gateway connector, the HTTP target connector, on the local gateway to send messages to the PeopleSoft listening connector on the remote gateway. Routing all messages through the local gateway enables each PeopleSoft Integration Broker system to keep its own centralized log of all integration messages.

Because this example shows two-way communication and assumes that the same service operation is being exchanged, the PeopleSoft Human Resources (USA) system and the PeopleSoft CRM (UK) system are source systems when they send messages, and they're destination systems when they receive messages.

Keep in mind the following as you review these configuration tasks:

- You should use a single integration gateway for all applications that reside on the same side of a firewall.
- The local integration gateway for one application is the remote integration gateway for the other application.

Copyright © 1988-2007, Oracle. All rights reserved.

Configuring the System for This Scenario

This section describes the configuration tasks for each of the components that are shown in the previous diagram.

This section discusses how to configure:

- The PeopleSoft Human Resources (USA) system.
- The PeopleSoft Human Resources (USA) integration gateway.
- The PeopleSoft CRM (UK) system.
- The PeopleSoft CRM (UK) integration gateway.

Configuring the PeopleSoft Human Resources (USA) System

On the PeopleSoft Human Resources (USA) system:

1. Define a local integration gateway.

   Use the Gateways component to define the local PeopleSoft Human Resources (USA) gateway.

2. Define a remote integration gateway.

   The remote integration gateway for the PeopleSoft Human Resources (USA) system is the PeopleSoft CRM (UK) gateway. Use the Gateways component to define a new gateway, and for the gateway URL, specify the PeopleSoft listening connector of the PeopleSoft CRM (UK) gateway.

3. Define the default local node.

   Use the Nodes component to define the default local node, which represents the PeopleSoft Human Resources (USA) system.

4. Define a remote node.

   The remote node that you define represents the PeopleSoft CRM (UK) system. When you set up the remote node, specify the PeopleSoft CRM (remote) integration gateway and the PeopleSoft target connector on that gateway.

5. For the outbound integration, define a service operation that specifies the request and response messages, the service operation handler definition, and the outbound routing definition.

   The outbound routing is a point-to-point routing where the PeopleSoft HR node is the sending node and the PeopleSoft CRM node is the receiving node.

6. For the inbound integration, on the same service operation create a routing definition where the PeopleSoft HR system is the receiving node and the PeopleSoft CRM system is the sending node.

   Note. The external alias you specify on the inbound routing definition must match the external alias on the outbound routing definition you created in the previous step.

Configuring the PeopleSoft Human Resources (USA) Integration Gateway

The only required integration gateway property that you must set for the PeopleSoft Human Resources (USA) integration gateway is the BEA Jolt connect strings that enable communication with the integration engine on the PeopleSoft Human Resources (USA) system. Set this property in the integrationGateway.properties file.

Configuring the PeopleSoft CRM (UK) System

On the PeopleSoft CRM (UK) system:

1. Define a local integration gateway.

Copyright © 1988-2007, Oracle. All rights reserved.

Use the Gateways component to define the local PeopleSoft CRM (UK) gateway.

2. Define a remote integration gateway.

   The remote integration gateway for the PeopleSoft CRM (UK) system is the PeopleSoft Human Resources (USA) gateway. Use the Gateways component to define a new gateway, and for the gateway URL, specify the PeopleSoft listening connector of the PeopleSoft Human Resources (USA) gateway.

3. Define the default local node.

   Use the Nodes component to define the default local node, which represents the PeopleSoft CRM (UK) system.

4. Define a remote node.

   The remote node that you define represents the PeopleSoft Human Resources (USA) system. When you set up the remote node, specify the PeopleSoft Human Resources (remote) integration gateway and the PeopleSoft target connector on that gateway.

5. For the outbound integration, define a service operation that specifies the request and response messages, the service operation handler definition, and the outbound routing definition.

   The outbound routing is a point-to-point routing where the PeopleSoft HR node is the receiving node and the PeopleSoft CRM node is the sending node.

6. For the inbound integration, on the same service operation create a routing definition where the PeopleSoft HR system is the sending node and the PeopleSoft CRM system is the receiving node.

Note. The external alias you specify on the inbound routing definition must match the external alias on the outbound routing definition you created in the previous step.

Configuring the PeopleSoft CRM (UK) Integration Gateway

The only required integration gateway property that you must set for the PeopleSoft CRM (UK) integration gateway is the BEA Jolt connect strings that enable communication with the integration engine on the PeopleSoft CRM (UK) system. Set this property in the integrationGateway.properties file.

See Also

Appendix A, “Integration Scenarios,” Understanding Integration Setup, page 679

Chapter 7, “Managing Integration Gateways,” Administering Integration Gateways, page 54

Chapter 15, “Managing Service Operations,” Adding Service Operation Definitions, page 298

Chapter 15, “Managing Service Operations,” Configuring Service Operation Definitions, page 298

Chapter 19, “Adding and Configuring Nodes,” page 361

Integrating with PeopleSoft Integration Broker Systems by Using Hubs

This section provides overviews of this scenario and hub routing types and discusses how to configure:

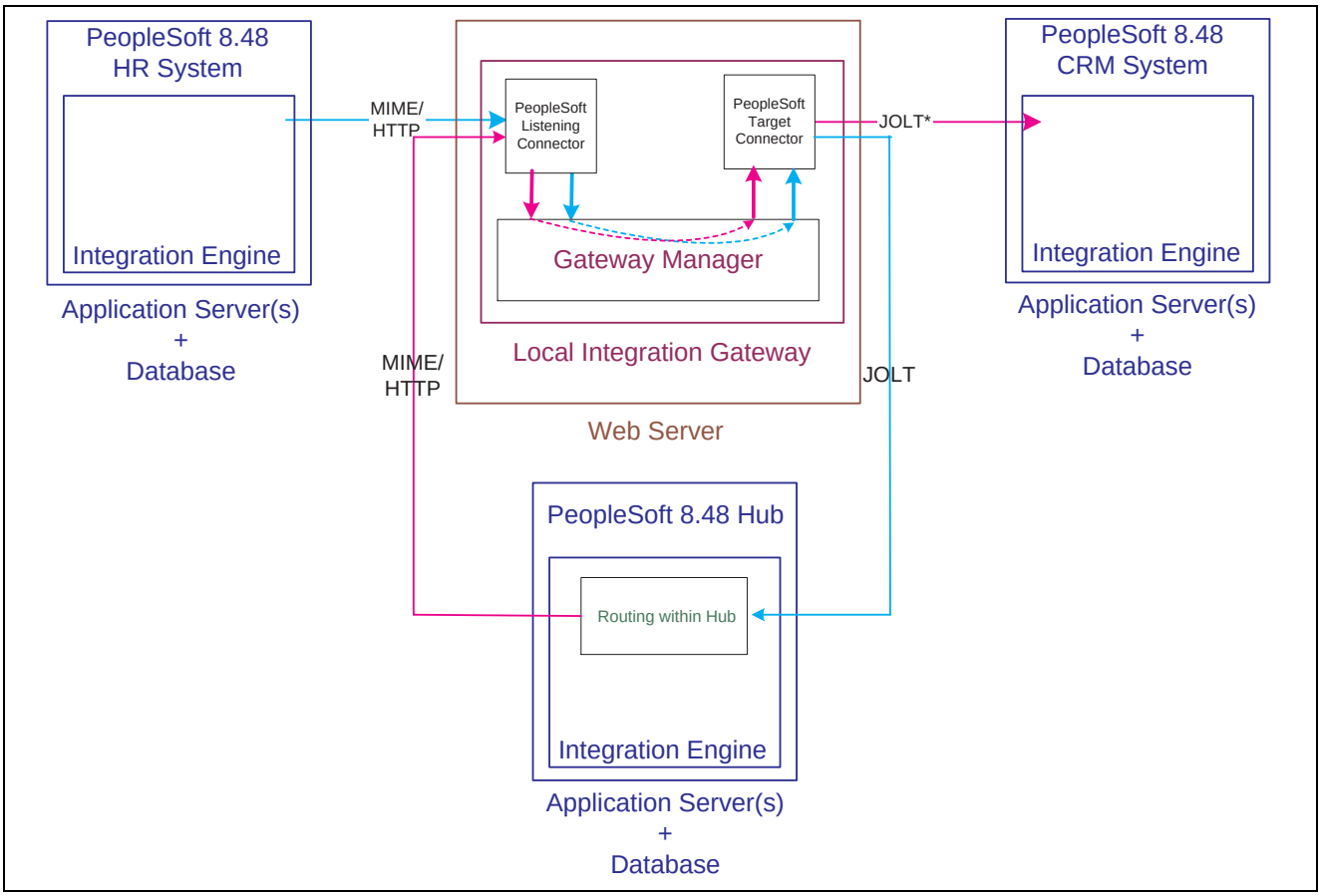
- Generic routing hubs.
- Sender-specified routing hubs.

 Copyright © 1988-2007, Oracle. All rights reserved.

Understanding This Scenario

A PeopleSoft Integration Broker hub configuration includes an integration engine that houses routing rules and transformations. All integrations are routed through the hub, which enables you to centralize routing rules and offload the transformation process.

This diagram shows a one-way hub configuration scenario that involves a PeopleSoft Human resources system and a PeopleSoft CRM system:



Integrations with PeopleSoft Integration Broker systems using hubs

In this scenario, all of the routing rules and transformations are located on the hub.

To implement integrations between the two systems without a hub, you must set up a complete set of complementary routing rules and transformations on each node.

Understanding Hub Routing Types

There are two hub routing types: generic routing and sender-specified routing. The configuration steps for a hub vary, depending on which routing type you choose.

With generic routing, all transactions from the participating systems are sent directly to a hub for routing and transformation.

Copyright © 1988-2007, Oracle. All rights reserved.

With sender-specified routing, a destination node name is passed as a parameter to a PeopleCode Publish or SyncRequest method, such as %intBrokerPublish or %intBroker.SynchRequest, to explicitly route the outbound transactions to the necessary node. Using sender-specified routing requires that you define the explicit destination nodes on the sending system. However, you can configure the system so that PeopleSoft Integration Broker passes these outbound transactions to the hub for possible rerouting and transformation.

Note. You must use sender-specific routing when you're using PublishXMLDoc to asynchronously publish an XML object.

Regardless of which hub routing you use, you must configure each PeopleSoft application's integration engine, the integration gateway, and the PeopleSoft Integration Broker hub. A PeopleSoft Integration Broker hub can be an installed PeopleSoft application, or it can have only a stand-alone PeopleTools database installed, which includes the integration engine.

Configuring Generic-Routing Hubs

By using the elements in the previous diagram as an example, this section provides an overview of how to configure a generic-routing hub. In the scenario, the PeopleSoft Human Resources system, the PeopleSoft CRM system, and the hub must all point to the same integration gateway and use the same gateway URL.

This section discusses how to configure:

- The PeopleSoft Human Resources system.
- The PeopleSoft CRM system.
- The PeopleSoft hub.
- The integration gateway.
- The generic-routing hub.

Configuring the PeopleSoft Human Resources System

On the PeopleSoft Human Resources system:

1. Define a local integration gateway.

   Use the Gateways component to set up a local integration gateway for sending messages.

2. Set up a local node.

   Use the Nodes component to set up the local node, which represents the PeopleSoft Human Resources system.

3. Set up a remote node.

   Use the Nodes component to set up the remote node, which represents the hub system.

4. Create a service operation.

   Use the Service Operations component to set up a service operation. Define the request message, service operation handler definition, and routing definition.

   Create a point-to-point routing definition where the PeopleSoft HR system is the sending node and the hub system is the receiving node.

Configuring the PeopleSoft CRM System

On the PeopleSoft CRM system:

 Copyright © 1988-2007, Oracle. All rights reserved.

1. Define a local integration gateway.

   Use the Gateways component to set up a local integration gateway for sending messages.

2. Set up a local node.

   Use the Node Definition component to set up the local node, which represents the PeopleSoft CRM system.

3. Set up a remote node.

   Use the Node Definition component to set up the remote node, which represents the hub system.

4. Create a service operation.

   Use the Service Operations component to set up a service operation.

   Create an inbound point-to-point routing definition where the sending node is the hub system and the receiving node is the PeopleSoft CRM system.

Configuring the PeopleSoft Hub

On the PeopleSoft hub:

1. Define a local integration gateway.

   Use the Gateways component to set up a local integration gateway for sending messages.

2. Set up a local node.

   Use the Node Definition component to set up the local node, which represents the hub system.

3. Set up remote nodes.

   Set up two remote nodes: one that represents the PeopleSoft Human Resources system and another that represents the PeopleSoft CRM system.

4. Create a service operation.

   Use the Service Operations component to set up a service operation.

   Create an outbound point-to-point routing definition where the sending node is the PeopleSoft HR system and the receiving node is the PeopleSoft CRM system.

Configuring the Integration Gateway

You must set integration gateway properties for the local gateway. The only required properties are the BEA Jolt connect string properties that enable communication with the integration engines on the PeopleSoft Human Resources, PeopleSoft CRM, and PeopleSoft hub systems. Set these properties in the integrationGateway.properties file.

Configuring the Generic-Routing Hub

For all messages going through the hub, you must set up a service operation and routing on the hub. By using the systems in the diagram as an example, the following table shows the node, service operation, and routing configurations that are required for generic routing through a hub:

Copyright © 1988-2007, Oracle. All rights reserved.

Item to Configure	PeopleSoft Human Resources System	Integration Broker Hub	PeopleSoft CRM System
Local nodes	Rename the default local node to represent the PeopleSoft Human Resources system.	Rename the default local node to represent the hub.	Rename the default local node to represent the PeopleSoft CRM system.
Remote nodes	Define a remote node to represent the hub system.	Define remote nodes to represent the PeopleSoft Human Resources and CRM systems.	Define a remote node to represent the hub.
Service operations and routings.	Define outbound routing to the hub system.	Create a service operation that contains an outbound point-to-point routing definition where the sending node is the PeopleSoft HR system and the receiving node is the PeopleSoft CRM system.	Define inbound routing from the hub system.

See Also

Appendix A, “Integration Scenarios,” Understanding Integration Setup, page 679

Chapter 7, “Managing Integration Gateways,” Administering Integration Gateways, page 54

Chapter 15, “Managing Service Operations,” Adding Service Operation Definitions, page 298

Chapter 15, “Managing Service Operations,” Configuring Service Operation Definitions, page 298

Chapter 19, “Adding and Configuring Nodes,” page 361

Configuring Sender-Specified Routing Hubs

By using the systems shown in the previous diagram as an example, this section provides an overview of how to configure a sender-specific routing hub. The PeopleSoft Human Resources system is the sending system and the PeopleSoft CRM system is the receiving system. In this scenario, the sending system, the receiving system, and the hub must all point to the same gateway and use the same gateway URL.

This section discusses how to configure:

- The PeopleSoft Human Resources (sending) system.
- PeopleSoft CRM (receiving) system.
- The PeopleSoft hub.
- The integration gateway.
- The sender-specified routing hub.

 Copyright © 1988-2007, Oracle. All rights reserved.

Configuring PeopleSoft Human Resources (Sending) System

On the PeopleSoft Human Resources system:

1. Define a local integration gateway.

   Use the Gateways component to set up a local integration gateway for sending messages.

2. Set up a local node.

   Use the Nodes component to set up the local node, which represents the PeopleSoft Human Resources system.

3. Set up remote nodes.

   Set up two remote nodes: one for the receiving system (PeopleSoft CRM in the example) and one for the hub. When setting up the PeopleSoft CRM remote node, on the Nodes-Node Definitions page in the Hub Node field, enter the node name of the hub system.

4. Create a service operation.

   Use the Service Operations component to create a service operation that contains an outbound point-to-point routing definition where the sending node is the PeopleSoft HR system and the receiving node is the PeopleSoft CRM system.

Configuring PeopleSoft CRM (Receiving System)

On the PeopleSoft CRM system:

1. Define a local integration gateway.

   Use the Gateways component to set up a local integration gateway for sending messages.

2. Set up a local node.

   Use the Nodes component to set up the local node, which represents the PeopleSoft CRM system.

3. Set up a remote node.

   Use the Nodes component to set up a node that represents the hub.

4. Create a service operation.

   Use the Service Operations component to create a service operation that contains an inbound point-to-point routing definition where the sending node is the hub system and the receiving node is the PeopleSoft CRM system.

5. Create a service operation.

   Use the Service Operations component to create a service operation that contains an inbound point-to-point routing definition where the sending node is the hub system and the receiving node is the PeopleSoft CRM system.

Configuring the PeopleSoft Hub

On the PeopleSoft hub:

1. Define a local integration gateway.

   Use the Gateways component to set up a local integration gateway for sending messages.

2. Set up a local node.

   Use the Nodes component to set up the local node, which represents the hub system.

3. Set up remote nodes.

Copyright © 1988-2007, Oracle. All rights reserved.

Use the Nodes component to set up two remote nodes: one for the PeopleSoft Human Resources system and one for the PeopleSoft CRM system.

4. Create a service operation.

   Use the Service Operations component to create a service operation that contains a point-to-point routing where the sending node is the PeopleSoft HR node and the receiving node is the CRM node.

Configuring the Integration Gateway

The only required integration gateway properties for the local integration gateway are the BEA Jolt connect string properties that enable communication with the integration engines on the target PeopleSoft Integration Broker systems. Set these properties in the integrationGateway.properties file.

Configuring the Sender-Specified Routing Hub

For all messages going through the hub, you must set up transactions and relationships on the hub. By using the systems in the previous diagram as example, the following table shows the node, transaction, and relationship configurations that are required for sender-specified routing through a hub from the PeopleSoft Human Resources system:

Item to Configure	PeopleSoft Human Resources System	PeopleSoft Integration Broker Hub	PeopleSoft CRM System
Local nodes	Rename the default local node to represent the PeopleSoft Human Resources system.	Rename the default local node to represent the hub.	Rename the default local node to represent the PeopleSoft CRM system.
Remote nodes	Define remote nodes to represent the PeopleSoft CRM and hub systems.	Define remote nodes to represent the PeopleSoft Human Resources and CRM systems.	Define a remote node to represent the hub.
Service operations and routings.	Create a service operation that contains an outbound point-to-point routing definition that specifies the receiving node is the PeopleSoft CRM system.	Create a service operation that contains a point-to-point routing where the sending node is the PeopleSoft HR node and the receiving node is the CRM node.	Create a service operation that contains an inbound point-to-point routing definition where the sending node is the hub system and the receiving node is the PeopleSoft CRM system.

All messages to the PeopleSoft CRM node are the result of publish statements, which include these target node parameters:

- msg.Publish(Node.CRM)
- SyncRequest(Node.CRM)
- %intBroker.publish(&MyDoc, Message.MyMessage, Node.CRM)
- %intBroker.SyncRequest(&MyDoc, Message.MyMessage, Node.CRM)

 Copyright © 1988-2007, Oracle. All rights reserved.

See Also

Appendix A, “Integration Scenarios,” Understanding Integration Setup, page 679

Chapter 7, “Managing Integration Gateways,” Administering Integration Gateways, page 54

Chapter 15, “Managing Service Operations,” Adding Service Operation Definitions, page 298

Chapter 15, “Managing Service Operations,” Configuring Service Operation Definitions, page 298

Chapter 19, “Adding and Configuring Nodes,” page 361

Integrating with Third-Party Systems

This section provides an overview of this scenario and discusses how to configure the system for this scenario.

Understanding This Scenario

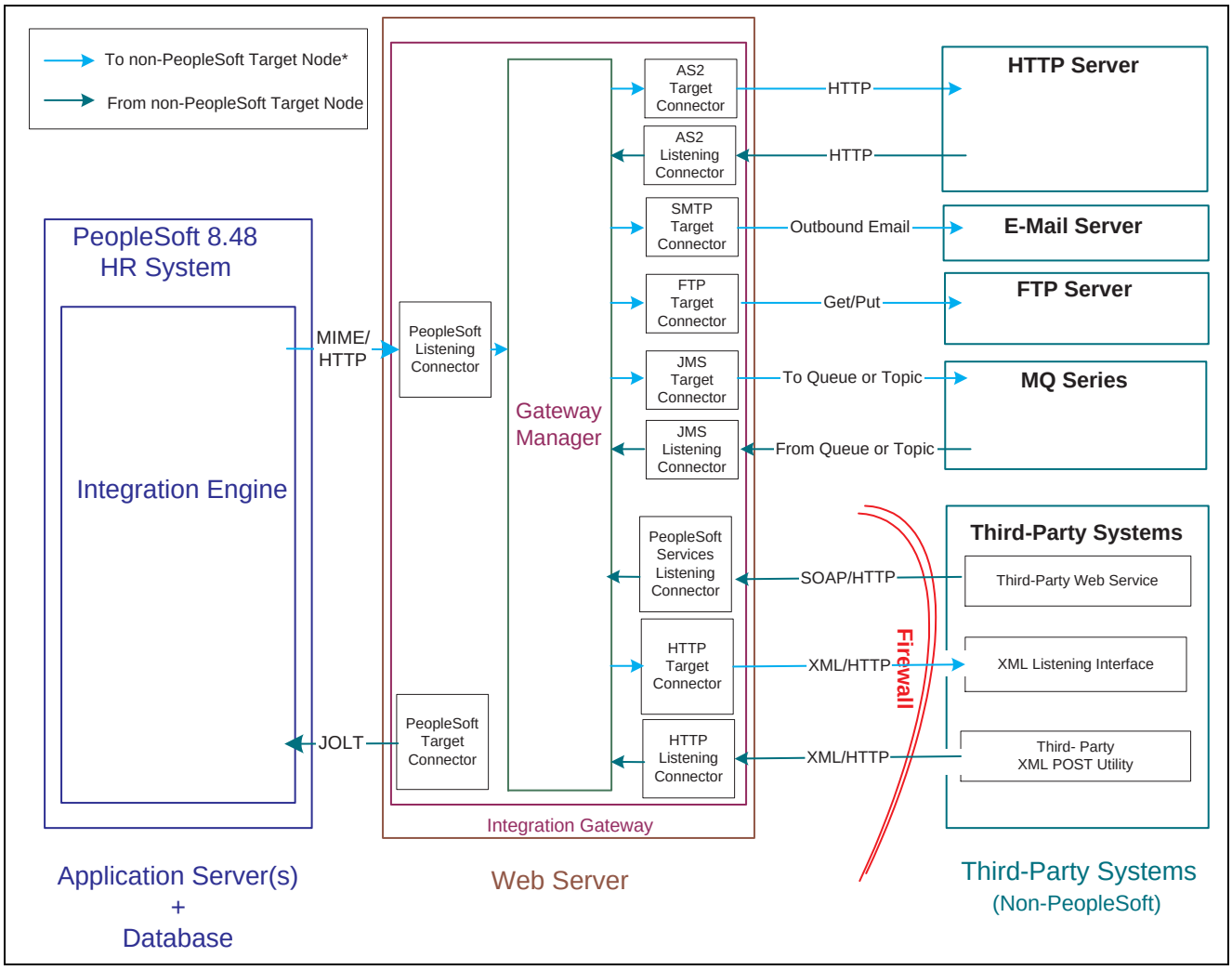
For communications with third-party systems, messages can go through local or remote gateways.

Sending a message to a third-party system is the same as sending a message to a PeopleSoft Integration Broker node, except that the target connector that you select depends on the third-party system with which you are communicating.

Messages from third-party systems can enter the gateway through any of the listening connectors that are delivered with PeopleSoft Integration Broker or through a listening connector that you build. You cannot use the PeopleSoft listening connector for integrations with third-party systems, because it can accept messages only in the PeopleSoft internal format.

This diagram shows the connectors that a PeopleSoft system can use to communicate with a third-party system and how the PeopleSoft system can communicate with third-party systems over a firewall:

Copyright © 1988-2007, Oracle. All rights reserved.



Integrations with third-party systems

Configuring the System for This Scenario

This section discusses how to configure:

- The PeopleSoft Human Resources system.
- The PeopleSoft Human Resources integration gateway.

Configuring the PeopleSoft Human Resources System

On the PeopleSoft Human Resources system:

1. Define a local integration gateway.

   Use the Gateways component to set up a local integration gateway for sending messages.

2. Create a service operation.

   Use the Service Operations component to create a service operation that contains an inbound point-to-point routing definition where the sending node is the hub system and the receiving node is the PeopleSoft CRM system.

3. Set up a remote node.

Copyright © 1988-2007, Oracle. All rights reserved.

Set up a remote node that represents the third-party system. When you define this node, you select the appropriate connector (for example, JMS target connector, SMTP target connector, and so forth) for communicating with the third-party system.

4. Create a service operation with an inbound routing definition where the sending node is the third-party system and the receiving node is the PeopleSoft HR system.
5. In the service operation definition you created in the previous step, create an outbound routing definition where the sending node is the PeopleSoft HR system and the receiving node is the third-party system.

Configuring the PeopleSoft Human Resources Integration Gateway

The only required integration gateway properties for the local integration gateway are the default BEA Jolt connect string properties that enable communication with integration engines on the PeopleSoft Human Resources system. Set these properties in the integrationGateway.properties file.

See Also

Appendix A, "Integration Scenarios," Understanding Integration Setup, page 679

Chapter 15, "Managing Service Operations," Adding Service Operation Definitions, page 298

Chapter 15, "Managing Service Operations," Configuring Service Operation Definitions, page 298

Chapter 19, "Adding and Configuring Nodes," page 361

Integrating with Third-Party Systems by Using Remote Gateways

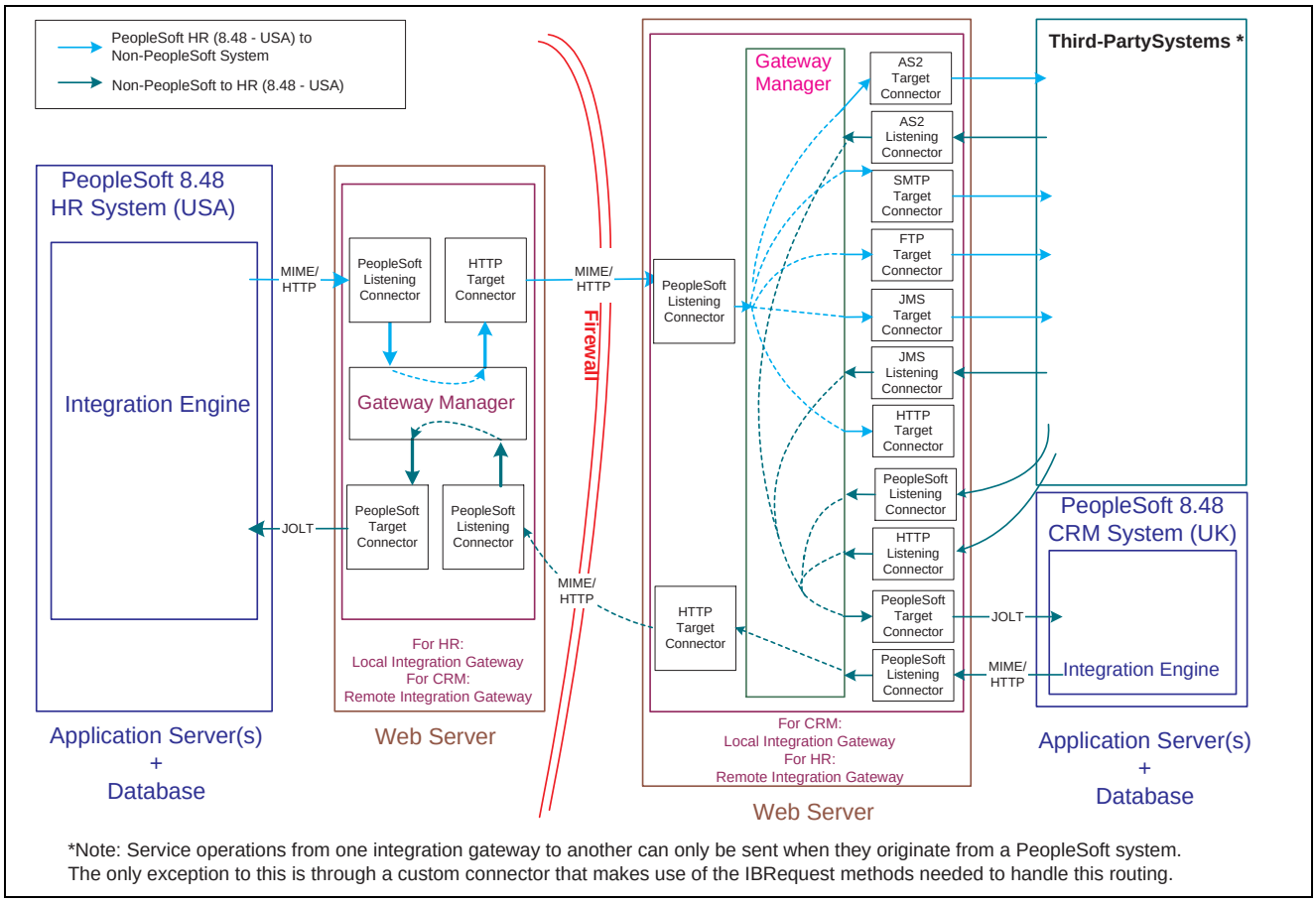
This section provides an overview of this scenario and discusses how to:

- Send messages to third-party systems.
- Receive messages from third-party systems.

Understanding This Scenario

This diagram shows how to integrate with third-party systems by using remote gateways:

Copyright © 1988-2007, Oracle. All rights reserved.



Integrations with third-party systems using remote gateways

Sending Messages to Third-Party Systems

This section discusses how to configure:

- The PeopleSoft Human Resources system.
- The PeopleSoft Human Resources integration gateway.

The setup for this scenario is similar to the configuration for integrations with Integration Broker systems. The only difference is the target connector you use.

However, instead of using the PeopleSoft target connector for the remote node you must select the target connector based on the third-party system with which the PeopleSoft system is communicating. The target connector you select must reside on the remote gateway.

In the previous diagram, the PeopleSoft Human Resources system is the source system and the selected target connector is shown on the other side of the firewall, on the remote integration gateway.

As you review the configuration tasks for this scenario, keep in mind the following points:

- PeopleSoft recommends using a single gateway for all applications that reside on one side of a firewall.
- The local gateway for one PeopleSoft application can be the remote gateway for the other PeopleSoft application.

Copyright © 1988-2007, Oracle. All rights reserved.

Configuring the PeopleSoft Human Resources System

On the PeopleSoft Human Resources system:

1. Define a local integration gateway.

   Use the Gateways component to define the local PeopleSoft Human Resources (USA) gateway.

2. Define a remote integration gateway on the local system.

   Use the Gateways component to define the gateway for the PeopleSoft Human Resources (USA) system (which is the PeopleSoft CRM [UK] gateway) and to specify the URL of the PeopleSoft CRM (UK) gateway.

3. Set up a local node.

   Use the Nodes component to set up the local node, which represents the PeopleSoft Human Resources (USA) system.

4. Set up a remote node.

   Use the Nodes component to set up the remote node, which represents the third-party system. When you define the remote node, use the Node Definitions-Connectors page to specify the gateway ID on the remote integration gateway. In addition, select the appropriate target connector, for example JMS target connector, SMTP target connector, POP 3 target connector, and so forth.

5. Create a service operation that includes an outbound routing definition where the sending node is the PeopleSoft HR system and the receiving node is the third-party system.

Configuring the PeopleSoft Human Resources Integration Gateway

The only required integration gateway properties are the BEA Jolt connect string properties that enable communication with integration engines on PeopleSoft Integration Broker systems. Set these properties in the integrationGateway.properties file.

See Also

Appendix A, “Integration Scenarios,” Understanding Integration Setup, page 679

Chapter 15, “Managing Service Operations,” Adding Service Operation Definitions, page 298

Chapter 15, “Managing Service Operations,” Configuring Service Operation Definitions, page 298

Chapter 19, “Adding and Configuring Nodes,” page 361

Receiving Messages from Third-Party Systems

The previous diagram shows a second configuration scenario where a third-party system is performing an inbound HTTP Post to a PeopleSoft Human Resources/USA system via a UK gateway. In this scenario, the message goes through the PeopleSoft CRM/UK system only to get routing information, before it is sent to the remote integration gateway (the gateway on the USA side of the firewall. Therefore, in this scenario, the PeopleSoft CRM/UK system serves as a hub.

This section discusses how to configure:

- The PeopleSoft Human Resources (USA) system.
- The PeopleSoft Human Resources integration gateway.
- The PeopleSoft CRM (UK) system and hub.
- The PeopleSoft CRM (UK) integration gateway.

Copyright © 1988-2007, Oracle. All rights reserved.

- The third-party system and PeopleSoft system.

Message Flow

In this scenario, a message originating from a third-party system is posted to the HTTP listening connector, JMS listening connector or a custom-built listening connector on the PeopleSoft CRM/UK integration gateway. Since the message does not contain the required routing information for the remote gateway, the listening connector hands it off to the PeopleSoft target connector. The PeopleSoft target connector sends the message to the default PeopleSoft node (the PeopleSoft CRM/UK) as determined by the default Jolt settings in the integrationGateway.properties file.

When the message reaches Integration Broker on the PeopleSoft CRM/UK system, the system applies transaction information to reroute the message to the remote gateway (the gateway on the USA side of the firewall), and thereby serves as a hub.

Message Processing on the Remote Gateway

Whenever you publish a message bound for a remote gateway, PeopleSoft Integration Broker reads it, determines that the target connector is not on its local gateway, places the remote gateway URL inside the IBInfo message wrapper and posts it to the PeopleSoft listening connector on the local gateway. The local gateway manager finds a remote gateway URL in the message wrapper and routes it to the remote gateway default connector, the HTTP target connector. The HTTP target connector on the local gateway then posts the message to the remote gateway URL (the PeopleSoft listening connector on the remote gateway) in MIME format, and removes the URL from the IBInfo message wrapper. On arrival at the remote gateway, the message is processed like any other incoming PeopleSoft message.

An exception to this message flow is if on the UK integration gateway side you created and loaded a custom listening connector that allows for the required routing information to be populated in the IBInfo message wrapper. The message would no longer need to be sent via the hub.

Keep in mind the following points:

- PeopleSoft recommends that you use a single gateway for all applications that reside on one side of a firewall.
- The local gateway for one PeopleSoft application can be a remote gateway for another PeopleSoft application.
- A message coming from a third-party system (local gateway or remote gateway) system can enter the integration gateway from any of the delivered listening connectors or from a custom-built listening connector. It cannot, however, use the PeopleSoft listening connector. PeopleSoft has designed the PeopleSoft listening connector to accept messages in the PeopleSoft internal message format only. Note that the diagram shows the message entering the integration gateway via the HTTP listening connector.

Configuring the PeopleSoft Human Resources (USA) System

On the PeopleSoft Human Resources system:

1. Set up a local node.

   Use the Node Definition component to set up the local node, which represents the Human Resources system.

2. Set up a remote node.

   The remote node that you set up represents the PeopleSoft CRM system. When you set up the remote node, specify the PeopleSoft target connector.

3. Create a service operation that contains an outbound routing definition where the sending node is the PeopleSoft CRM system and the receiving node is the PeopleSoft HR system.

 Copyright © 1988-2007, Oracle. All rights reserved.

Configuring the PeopleSoft Human Resources Integration Gateway

The only required integration gateway properties are the BEA Jolt connect string properties that enable communication with the PeopleSoft Human Resources system. Set these properties in the integrationGateway.properties file.

Configuring the PeopleSoft CRM (UK) System and Hub

On the PeopleSoft CRM (UK) system:

- Define a local integration gateway.

  Use the Gateways component to set up the local gateway for the sending system.

- Define a remote integration gateway.

  The remote gateway for the PeopleSoft CRM (UK) system is the PeopleSoft Human Resources (USA) gateway.

- Set up a local node.

  Use the Node Definition component to set up the local node, which represents the PeopleSoft CRM system.

- Set up remote nodes.

  Use the Node Definition component to define two remote nodes: one remote node that represents the third-party system and another to represent the PeopleSoft Human Resources (USA) system. When you define the remote node that represents the third-party system, you specify the HTTP target connector, HTTPTARGET. When you define the remote node that represents the PeopleSoft Human Resources (USA) system, set it to use the PeopleSoft target connector on the remote (USA) gateway.

- Create a service operation that contains an outbound routing definition where the sending node is the third-party system and the receiving node is the PeopleSoft HR system.

- In the service operation definition that you created in the previous step, create an inbound routing definition if the third-party system will be sending you requests.

Configuring the PeopleSoft CRM (UK) Integration Gateway

The only required properties are the BEA Jolt connect string properties that enable communication with integration engines on the PeopleSoft CRM systems. Set these properties in the integrationGateway.properties file.

Configuring the Third-Party System and PeopleSoft System

Because the PeopleSoft CRM (UK) system serves as a hub in this scenario, you must set up transactions and relationships for all messages from the third-party system that are routed through it. By using the systems in the diagram as an example, the following table shows the required node, transaction and relationship configurations:

Copyright © 1988-2007, Oracle. All rights reserved.

Item to Configure	PeopleSoft Human Resources System	PeopleSoft CRM System (Hub)
Local nodes	Rename the default local node to represent the PeopleSoft Human Resources system.	Rename the default local node to represent the PeopleSoft CRM system.
Remote nodes	Define a remote node to represent the PeopleSoft CRM system.	Define remote nodes to represent the third-party system and the PeopleSoft Human Resources system.
Service operations and routing definitions	Define a service operation with an inbound routing definition where the PeopleSoft CRM system is the sending node and the PeopleSoft HR system is the receiving node.	Define a service operation with an outbound routing definition where the sending node is the third-party system and the receiving node is the PeopleSoft HR system.

See Also

Appendix A, "Integration Scenarios," Understanding Integration Setup, page 679

Chapter 7, "Managing Integration Gateways," Administering Integration Gateways, page 54

Chapter 15, "Managing Service Operations," Adding Service Operation Definitions, page 298

Chapter 15, "Managing Service Operations," Configuring Service Operation Definitions, page 298

Chapter 19, "Adding and Configuring Nodes," page 361

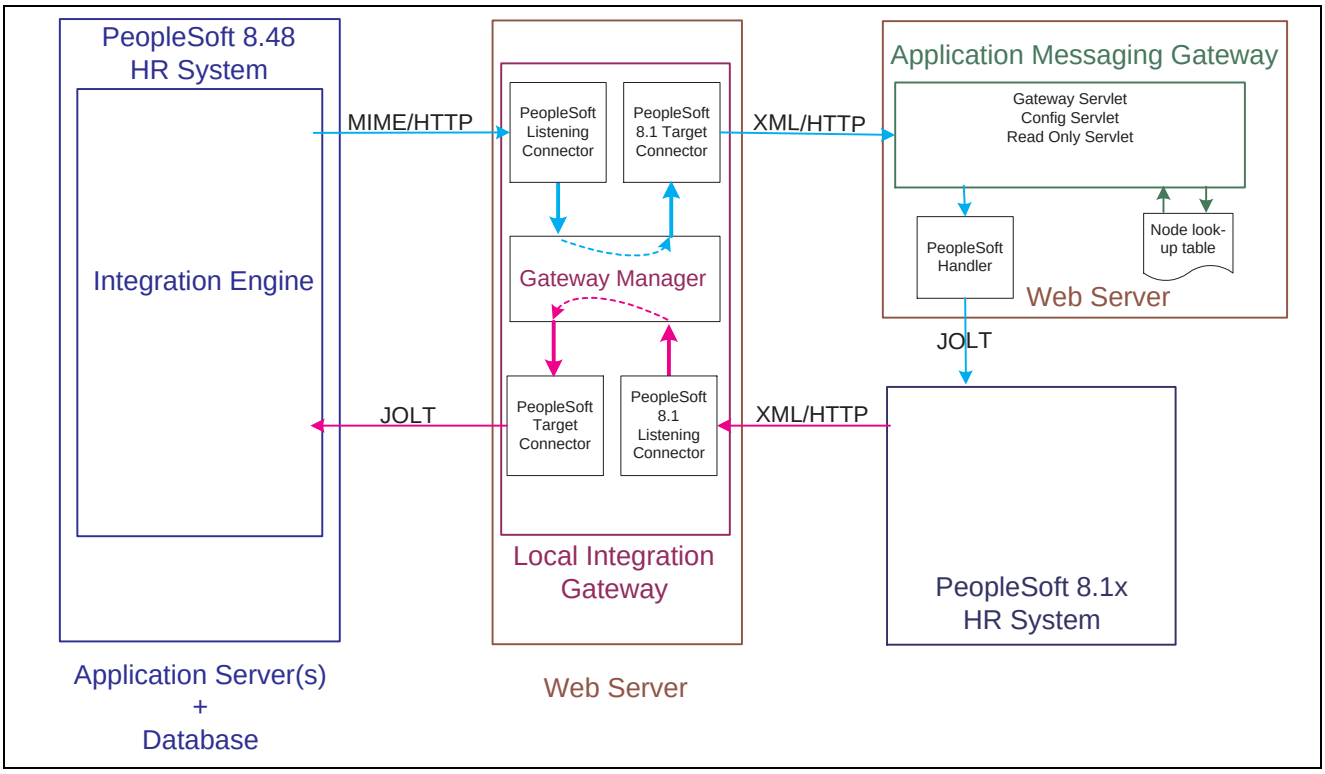
Integrating with PeopleSoft 8.1x Systems

This section provides an overview of this scenario and discusses how to configure the system for this scenario.

Understanding This Scenario

This diagram shows the PeopleSoft Integration Broker components and configuration for communications between PeopleSoft Integration Broker systems and PeopleSoft 8.1x systems:

 Copyright © 1988-2007, Oracle. All rights reserved.



Integrations with PeopleSoft 8.1x systems

In this scenario, you must configure the PeopleSoft Integration Broker system, the integration gateway, and the PeopleSoft 8.1x system. The remainder of this section highlights these tasks by using the systems and components shown in the diagram as examples.

Configuring the System for This Scenario

This section discusses how to configure:

- The PeopleSoft Human Resources system.
- The PeopleSoft Human Resources integration gateway.
- The PeopleSoft 8.1x Human Resources system.

Configuring the PeopleSoft Human Resources System

On the PeopleSoft Human Resources system:

- Define a local integration gateway.

  Use the Gateways component to set up a local gateway for the PeopleSoft Human Resources system.

- Set up a local node.

  Use the Node Definition component to set up the local node, which represents the PeopleSoft Human Resources system.

- Set up a remote node.

  The remote node that you set up represents the PeopleSoft 8.1x Human Resources system. When you set up the remote node, specify the PeopleSoft 8.1 target connector (PSFT81TARGET) on the Connectors tab.

Copyright © 1988-2007, Oracle. All rights reserved.

Note. If you have upgraded from a PeopleSoft 8.1x system, all nodes that existed for the system have been preserved as remote nodes in the PeopleSoft Integration Broker system. However, you must then associate each of these nodes to the PeopleSoft 8.1 target connector.

- Create a service operation with an inbound routing definition.

  Use the Service Operations component to create a service operation that contains an inbound routing definition where the receiving node is the PeopleSoft 8.49 system and the sending node is the PeopleSoft 8.1x system.

- Set up an outbound routing definition.

  In the service operation definition that you created in the previous step, create and outbound routing definition where the sending node is the PeopleSoft 8.49 system and the receiving node is the 8.1x system.

Configuring the PeopleSoft Human Resources Integration Gateway

You must set integration gateway properties for the local gateway. The only required properties are the BEA Jolt connect string properties that enable communication with the PeopleSoft Human Resources systems. Set these properties in the integrationGateway.properties file.

Configuring the PeopleSoft 8.1x Human Resources System

On the PeopleSoft 8.1x Human Resources system, locate the PeopleSoft 8.1x Human Resources message node and change the URL (location) to the PeopleSoft listening connector. The format is http://*webserver*/PSIGW/PS81ListeningConnector.

See Also

Appendix A, “Integration Scenarios,” Understanding Integration Setup, page 679

Chapter 7, “Managing Integration Gateways,” Administering Integration Gateways, page 54

Chapter 15, “Managing Service Operations,” Adding Service Operation Definitions, page 298

Chapter 15, “Managing Service Operations,” Configuring Service Operation Definitions, page 298

Chapter 19, “Adding and Configuring Nodes,” page 361

 Copyright © 1988-2007, Oracle. All rights reserved.

APPENDIX B

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 meta data 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 the 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.

Copyright © 1988-2007, Oracle. All rights reserved.

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*

 Copyright © 1988-2007, Oracle. All rights reserved.

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:

Copyright © 1988-2007, Oracle. All rights reserved.

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.

 Copyright © 1988-2007, Oracle. All rights reserved.

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 Chapter 22, "Managing Error Handling, Logging, Tracing, and Debugging," Managing Integration Gateway Message and Error Logging, page 499.

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 8.49 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:

Copyright © 1988-2007, Oracle. All rights reserved.

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.

 Copyright © 1988-2007, Oracle. All rights reserved.

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 txt 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.

Copyright © 1988-2007, Oracle. All rights reserved.

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*.

 Copyright © 1988-2007, Oracle. All rights reserved.

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;
```

Copyright © 1988-2007, Oracle. All rights reserved.

```
/* 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>
   ```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
      <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=
   ```

Copyright © 1988-2007, Oracle. All rights reserved.

```
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>
```

 Copyright © 1988-2007, Oracle. All rights reserved.

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:

Copyright © 1988-2007, Oracle. All rights reserved.

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 8.49 node and a PeopleSoft 8.1 node.

Understanding the PeopleSoft 8.1 Connectors Examples

When sending a message from a PeopleSoft 8.49 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 8.49 System

To set up data on the PeopleSoft 8.49 system:

 Copyright © 1988-2007, Oracle. All rights reserved.

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 8.49 system.
9. Add an outbound routing for the PSFT81_SERVICE_OPR service operation with the sourcenode being the 8.49 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();
```

Copyright © 1988-2007, Oracle. All rights reserved.

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 8.49 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 8.49 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 8.49 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.

 Copyright © 1988-2007, Oracle. All rights reserved.

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 8.49 system to a PeopleSoft 8.1 system.

To send a message from a PeopleSoft 8.49 system to a PeopleSoft 8.1 system:

1. In the PeopleSoft Pure Internet Architecture on the PeopleSoft 8.49 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 8.49 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 8.49 system.

To send a message from a PeopleSoft 8.1 system to a PeopleSoft 8.49 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 8.49 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.

Copyright © 1988-2007, Oracle. All rights reserved.

See Also

Chapter 9, "Using Listening Connectors and Target Connectors," Working With the JMS Connectors, page 141

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 8.49 PeopleBook: PeopleSoft 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);
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
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);
```

Copyright © 1988-2007, Oracle. All rights reserved.

```
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 9, “Using Listening Connectors and Target Connectors,” Using the JMS Listening Connector, page 142.

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>
      ```

 Copyright © 1988-2007, Oracle. All rights reserved.

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 9, "Using Listening Connectors and Target Connectors," Working With the AS2 Connectors, page 164

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 27, "Setting Up Secure Integration Environments," page 583

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 */
```

Copyright © 1988-2007, Oracle. All rights reserved.

```
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 <PS_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)
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
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 XmlNode &rootNode, &descNode;

   /* get the body of the incoming message */
   &xmlDoc = GetMessageXmlDoc(); /* and write it to a file */
```

Copyright © 1988-2007, Oracle. All rights reserved.

```
&theFile = GetFile("AS2Request.txt", "W");
&theFile.WriteString(&xmlDoc.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>
```

 Copyright © 1988-2007, Oracle. All rights reserved.

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_HOME>\appserv\<APPSERVER_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);
```

Copyright © 1988-2007, Oracle. All rights reserved.

```
&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/>";
   ```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
/* 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.

Copyright © 1988-2007, Oracle. All rights reserved.

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>
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
<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.

Copyright © 1988-2007, Oracle. All rights reserved.

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>
```


Copyright © 1988-2007, Oracle. All rights reserved.

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.

Copyright © 1988-2007, Oracle. All rights reserved.

 Copyright © 1988-2007, Oracle. All rights reserved.

APPENDIX C

Transformation Example: Integration Between Two PeopleSoft Nodes

This appendix discusses how to:

- Create message definitions.
- Set up codesets.
- Set up transformations.
- Walk through the generated XSL code.
- Test the transformation.

Understanding This Appendix

This appendix presents an in-depth example of how to use transformations to alter the messages sent between two systems.

Using the Example

The purpose of this appendix is to present a more in depth example of how transformations can be used to alter the messages sent between two systems.

The following example describes an integration between a PeopleSoft SCM node and a PeopleSoft CRM node. This example demonstrates taking a PeopleSoft SCM purchase order message and transforming it into a similar PeopleSoft CRM purchase order message format.

Two aspects of transformations will be examined in this example:

- How XSL can be used to modify the structure of a message.
- How codesets can be used to map values between messages.

The example will focus on the PeopleSoft SCM system. This is the node where the XSL will be executed to transform the message from the SCM specified format to that of CRM. When the SCM system sends the message to CRM, the message will be in the CRM native format.

Integration Metadata for This Example

The example does not go into detail about setting up the all infrastructure necessary to actually send messages between the two nodes. It assumes that the systems have been configured and the service operations and services have been defined. Please refer to the chapters elsewhere in this PeopleBook for setting up integration metadata.

Copyright © 1988-2007, Oracle. All rights reserved.

See Also

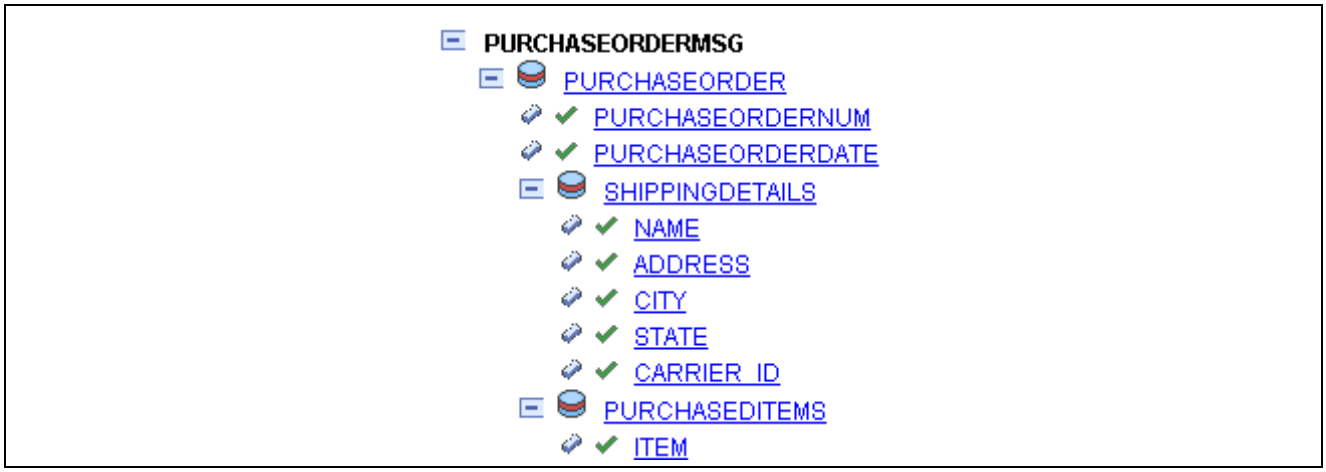
Chapter 4, "Understanding Creating and Implementing Integrations," Creating Integration Metadata, page 35

Creating Message Definitions

This section discusses the structure of the message definitions used in this example.

Message Definition: PeopleSoft SCM Node

The following example shows the format of the purchase order message on the PeopleSoft SCM node.



PURCHASEORDERMSG message definition.

The following is a sample message that corresponds to the message structure:

```
<?xml version="1.0"?>
<PURCHASEORDERMSG>
  <FieldTypes>
    <PURCHASEORDER class="R">
      <PURCHASEORDERNUM type="CHAR"/>
      <PURCHASEORDERDATE type="DATE"/>
    </PURCHASEORDER>
    <SHIPPINGDETAILS class="R">
      <NAME type="CHAR"/>
      <ADDRESS type="CHAR"/>
      <CITY type="CHAR"/>
      <STATE type="CHAR"/>
      <CARRIER_ID type="CHAR"/>
    </SHIPPINGDETAILS>
    <PURCHASEDITEMS class="R">
      <ITEM type="CHAR"/>
    </PURCHASEDITEMS>
    <PSCAMA class="R">
      <LANGUAGE_CD type="CHAR"/>
      <AUDIT_ACTN type="CHAR"/>
```

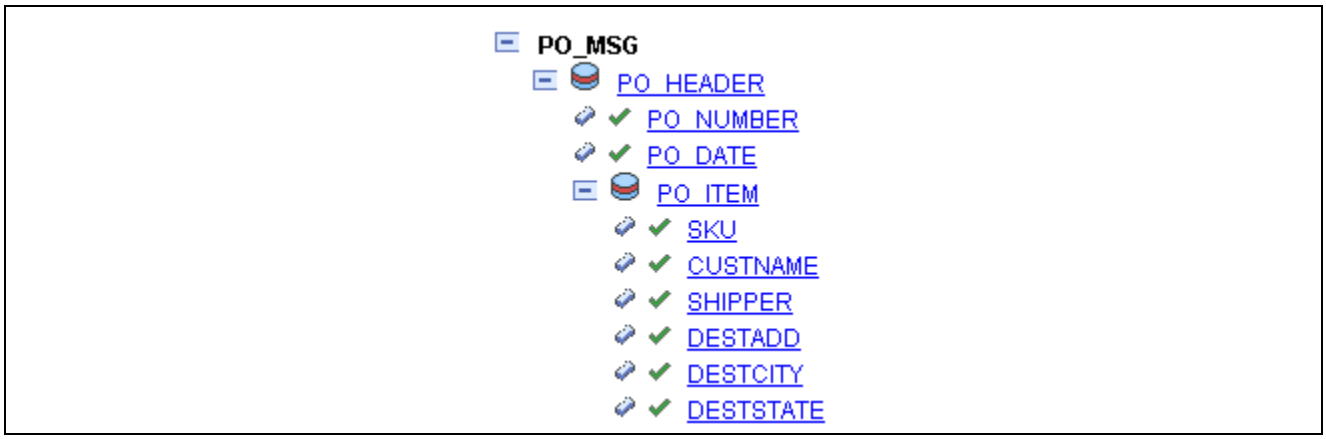
 Copyright © 1988-2007, Oracle. All rights reserved.

```
        <BASE_LANGUAGE_CD type="CHAR"/>
        <MSG_SEQ_FLG type="CHAR"/>
        <PROCESS_INSTANCE type="NUMBER"/>
        <PUBLISH_RULE_ID type="CHAR"/>
        <MSGNODENAME type="CHAR"/>
      </PSCAMA>
    </FieldTypes>
    <MsgData>
      <Transaction>
        <PURCHASEORDER class="R">
          <PURCHASEORDERNUM IsChanged="Y">19908</PURCHASEORDERNUM>
          <PURCHASEORDERDATE IsChanged="Y">2006-04-03</PURCHASEORDERDATE>
          <SHIPPINGDETAILS class="R">
            <NAME IsChanged="Y">Smith,Bill</NAME>
            <ADDRESS IsChanged="Y">123 Anywhere St</ADDRESS>
            <CITY IsChanged="Y">Fresno</CITY>
            <STATE IsChanged="Y">CA</STATE>
            <CARRIER_ID IsChanged="Y">USPS</CARRIER_ID>
          </SHIPPINGDETAILS>
          <PURCHASEDITEMS class="R">
            <ITEM IsChanged="Y">AAS5536</ITEM>
          </PURCHASEDITEMS>
          <PURCHASEDITEMS class="R">
            <ITEM IsChanged="Y">POB332Q</ITEM>
          </PURCHASEDITEMS>
        </PURCHASEORDER>
        <PSCAMA class="R">
          <LANGUAGE_CD>ENG</LANGUAGE_CD>
          <AUDIT_ACTN/>
          <BASE_LANGUAGE_CD>ENG</BASE_LANGUAGE_CD>
          <MSG_SEQ_FLG/>
          <PROCESS_INSTANCE>0</PROCESS_INSTANCE>
          <PUBLISH_RULE_ID/>
          <MSGNODENAME/>
        </PSCAMA>
      </Transaction>
    </MsgData>
  </PURCHASEORDERMSG>
```

Message Definition: PeopleSoft CRM Node

The following example shows the format of the purchase order on the PeopleSoft CRM node.

Copyright © 1988-2007, Oracle. All rights reserved.



PO_MSG message definition.

This is a sample message that corresponds to the message structure:

```
<?xml version="1.0"?>
<PO_MSG>
  <FieldTypes>
    <PO_HEADER class="R">
      <PO_NUMBER type="CHAR"/>
      <PO_DATE type="DATE"/>
    </PO_HEADER>
    <PO_ITEM class="R">
      <SKU type="CHAR"/>
      <CUSTNAME type="CHAR"/>
      <SHIPPER type="CHAR"/>
      <DESTADD type="CHAR"/>
      <DESTCITY type="CHAR"/>
      <DESTSTATE type="CHAR"/>
    </PO_ITEM>
    <PSCAMA class="R">
      <LANGUAGE_CD type="CHAR"/>
      <AUDIT_ACTN type="CHAR"/>
      <BASE_LANGUAGE_CD type="CHAR"/>
      <MSG_SEQ_FLG type="CHAR"/>
      <PROCESS_INSTANCE type="NUMBER"/>
      <PUBLISH_RULE_ID type="CHAR"/>
      <MSGNODENAME type="CHAR"/>
    </PSCAMA>
  </FieldTypes>
  <MsgData>
    <Transaction>
      <PO_HEADER class="R">
        <PO_NUMBER IsChanged="Y">BBN7782</PO_NUMBER>
        <PO_DATE IsChanged="Y">2006-04-15</PO_DATE>
        <PO_ITEM class="R">
          <SKU IsChanged="Y">JN557BB</SKU>
          <CUSTNAME IsChanged="Y">Jones,Mark</CUSTNAME>
          <SHIPPER IsChanged="Y">Federal Express</SHIPPER>
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
            <DESTADD IsChanged="Y">66  Availer St</DESTADD>
            <DESTCITY IsChanged="Y">Stockton</DESTCITY>
            <DESTSTATE IsChanged="Y">CA</DESTSTATE>
          </PO_ITEM>
        </PO_HEADER>
        <PSCAMA class="R">
          <LANGUAGE_CD>ENG</LANGUAGE_CD>
          <AUDIT_ACTN/>
          <BASE_LANGUAGE_CD>ENG</BASE_LANGUAGE_CD>
          <MSG_SEQ_FLG/>
          <PROCESS_INSTANCE>0</PROCESS_INSTANCE>
          <PUBLISH_RULE_ID/>
          <MSGNODENAME/>
        </PSCAMA>
      </Transaction>
    </MsgData>
</PO_MSG>
```

Setting Up the Codesets

Both the PeopleSoft SCM and PeopleSoft CRM messages contain an entry to capture the shipping agent to be used for the purchase order. For the PeopleSoft SCM message, the element used is CARRIER_ID; for the PeopleSoft CRM message it is SHIPPER. For the purposes of this example, the assumption is that the values for these two elements map differently, according to the following table:

CARRIER_ID values on SCM System	SHIPPER Values on CRM System
FEDEX	Federal Express
UPS	United Parcel Service
USPS	United States Postal Service

An SCM message containing the following XML:

```
<CARRIER_ID>UPS</CARRIER_ID>
```

would logically map to

```
<SHIPPER>United Parcel Service</SHIPPER>
```

in the corresponding CRM message.

In order to enable such mapping in this example, you must create codeset metadata.

Create codeset metadata:

1. Create a codeset group called *SCM_GROUP*. Add the following entries:

Copyright © 1988-2007, Oracle. All rights reserved.

Match Name	Match Value
CARRIER_ID	FEDEX
CARRIER_ID	UPS
CARRIER_ID	USPS
CARRIER_ID	Blank

Note that the final match value entry is blank: this will be used for the default value.

2. Create a codeset called *SCM_CODESET*, using the codeset group defined in the prior step. Add the following entries to this codeset:

Match Name	Match Value
CARRIER_ID	FEDEX
CARRIER_ID	UPS
CARRIER_ID	USPS
CARRIER_ID	Blank

3. Create a codeset group called *CRM_GROUP*. There is no need to define any entries for this group.
4. Add codeset values from the *SCM_GROUP* to the *CRM_GROUP,* using the *SCM_CODESET*. Four entries will need to be defined:
   - Select *CARRIER_ID* and <*blank*>, and add a return name of *SHIPPER* and the value *Unknown*.
   - Select *CARRIER_ID* and *FEDEX,* and add a return name of *SHIPPER* and the value *Federal Express*.
   - Select *CARRIER_ID* and *UPS*, and add a return name of *SHIPPER* and the value *United Parcel Service*.
   - Select *CARRIER_ID* and *USPS*, and add a return name of *SHIPPER* and the value *United States Postal Service*.
5. Go to the node definition for the SCM node and add the codeset group *SCM_CODESET* to the node.
6. Go to the node definition for the CRM node and add the codeset group *CRM_CODESET* to the node.

Setting Up the Transformation

The PeopleSoft SCM and PeopleSoft CRM purchase order messages share what are essentially the same first two elements: a purchase order number and the purchase order date. However, from there the structures differ. The PeopleSoft SCM message has a small section containing shipping information, followed by the list of items purchased. The PeopleSoft CRM message has the shipping information merged with the list of items purchased.

The following XSL transforms the PeopleSoft SCM purchase order message into the PeopleSoft CRM format:

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

 Copyright © 1988-2007, Oracle. All rights reserved.

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


  <xsl:template match="PURCHASEORDERMSG">
    <PO_MSG>
      <xsl:apply-templates select="FieldTypes"/>
      <xsl:apply-templates select="MsgData"/>
    </PO_MSG>
  </xsl:template>

  <xsl:template match="MsgData">
    <MsgData>
      <xsl:apply-templates select="Transaction"/>
    </MsgData>
  </xsl:template>

  <xsl:template match="Transaction">
    <Transaction>
      <xsl:apply-templates select="PURCHASEORDER"/>
      <PSCAMA class="R">
        <LANGUAGE_CD>ENG</LANGUAGE_CD>
        <AUDIT_ACTN/>
        <BASE_LANGUAGE_CD>ENG</BASE_LANGUAGE_CD>
        <MSG_SEQ_FLG/>
        <PROCESS_INSTANCE>0</PROCESS_INSTANCE>
        <PUBLISH_RULE_ID/>
        <MSGNODENAME/>
      </PSCAMA>
    </Transaction>
  </xsl:template>

  <xsl:template match="PURCHASEORDER">
    <PO_HEADER class="R">
      <PO_NUMBER IsChanged="Y"><xsl:value-of select="PURCHASEORDERNUM"/>
      </PO_NUMBER>
      <PO_DATE IsChanged="Y"><xsl:value-of select="PURCHASEORDERDATE"/>
      </PO_DATE>
      <xsl:apply-templates select="PURCHASEDITEMS"/>
    </PO_HEADER>
  </xsl:template>

  <xsl:template match="PURCHASEDITEMS">
    <PO_ITEM class="R">
      <SKU IsChanged="Y"><xsl:value-of select="ITEM"/></SKU>
      <CUSTNAME IsChanged="Y">
        <xsl:value-of select="../SHIPPINGDETAILS/NAME"/>
      </CUSTNAME>
      <SHIPPER IsChanged="Y">
         <psft_function name="codeset" codesetname="SCM_CODESET">
```

Copyright © 1988-2007, Oracle. All rights reserved.

```
            <parm name="CARRIER_ID">
               <xsl:value-of select="../SHIPPINGDETAILS/CARRIER_ID"/>
            </parm>
            <value name="SHIPPER" select="."/>
         </psft_function>
 </SHIPPER>
    <DESTADD IsChanged="Y">
      <xsl:value-of select="../SHIPPINGDETAILS/ADDRESS"/>
    </DESTADD>
    <DESTCITY IsChanged="Y">
      <xsl:value-of select="../SHIPPINGDETAILS/CITY"/>
    </DESTCITY>
    <DESTSTATE IsChanged="Y">
      <xsl:value-of select="../SHIPPINGDETAILS/STATE"/>
    </DESTSTATE>
  </PO_ITEM>
</xsl:template>

<xsl:template match="FieldTypes">
<FieldTypes>
  <PO_HEADER class="R">
    <PO_NUMBER type="CHAR"/>
    <PO_DATE type="DATE"/>
  </PO_HEADER>
  <PO_ITEM class="R">
    <SKU type="CHAR"/>
    <CUSTNAME type="CHAR"/>
    <SHIPPER type="CHAR"/>
    <DESTADD type="CHAR"/>
    <DESTCITY type="CHAR"/>
    <DESTSTATE type="CHAR"/>
  </PO_ITEM>
  <PSCAMA class="R">
    <LANGUAGE_CD type="CHAR"/>
    <AUDIT_ACTN type="CHAR"/>
    <BASE_LANGUAGE_CD type="CHAR"/>
    <MSG_SEQ_FLG type="CHAR"/>
    <PROCESS_INSTANCE type="NUMBER"/>
    <PUBLISH_RULE_ID type="CHAR"/>
    <MSGNODENAME type="CHAR"/>
  </PSCAMA>
</FieldTypes>
</xsl:template>
</xsl:stylesheet>
```

Note that this XSL contains a reference to the psft_function, which will resolve codeset mapping after the transform has been run.

This XSL should be placed into an Application Engine program, and this program associated with the routing definitions for the service operation.

 Copyright © 1988-2007, Oracle. All rights reserved.

XSL Walkthrough

This section discusses the process flow through the XSL when the PeopleSoft SCM purchase order message is transformed.

XSL is composed of templates. An XSLT template is an instruction of what to do when a particular XML node or condition is encountered. During a transform, the XSL processing engine starts at the root element of the XML message and then attempts to find matching templates in the XSL. When a matching template is found, the instructions in that template are used in the building of the output XML document.

The processing of a transform is actually a two pass process. In the first pass, the XSL is executed against the input XML, and an output XML document is generated. In the second pass, the psft_function calls are resolved and the codeset values are placed into the document.

Transformation Processing: First Pass

This section discusses the first pass of transform processing, during which the XSL code runs against the input XML and the system generates an output XML document. The steps listed below mimic the actions taken by the transformation engine in processing the input XML.

1. The first element in the input message is *PURCHASEORDERMESSAGE*. The transformation engine finds the following matching template in the XSL:

```
<xsl:template match="PURCHASEORDERMSG">
   <PO_MSG>
     <xsl:apply-templates select="FieldTypes"/>
     <xsl:apply-templates select="MsgData"/>
   </PO_MSG>
 </xsl:template>
```

   Here we see the root element of the output document being created. This template instructs the transformation engine to:

   - Output the start tag for *PO_MSG*.

     It is important to note that template processing is a recursive process. In the sequence above, the transformation engine outputs the start tag, then applies templates to all FieldType nodes. This is essentially a callout to the template that handles those nodes (and potentially any children). The output of this call is then placed into the output document. Then the transformation engine selects all the MsgData nodes, and applies templates to them. Again, the output from the processing of those nodes is then place into the output document. Finally, the closing PO_MSG tag is written into the output, and the first pass is finished. Of course, documenting a recursive process is not always straightforward. In this example bear in mind that while it is presented as a numbered sequence of steps, the actual process is not sequential.

   - Select the FieldTypes nodes under the current node in the input document, and process them.
   - Select the MsgData nodes under the current node in the input document, and process them.
   - Output the end tag for PO_MSG.

2. The transform engine selects the FieldTypes node, and finds the following template:

```
<xsl:template match="FieldTypes">
 <FieldTypes>
   <PO_HEADER class="R">
```

Copyright © 1988-2007, Oracle. All rights reserved.

```
      <PO_NUMBER type="CHAR"/>
      <PO_DATE type="DATE"/>
    </PO_HEADER>
    <PO_ITEM class="R">
      <SKU type="CHAR"/>
      <CUSTNAME type="CHAR"/>
      <SHIPPER type="CHAR"/>
      <DESTADD type="CHAR"/>
      <DESTCITY type="CHAR"/>
      <DESTSTATE type="CHAR"/>
    </PO_ITEM>
    <PSCAMA class="R">
      <LANGUAGE_CD type="CHAR"/>
      <AUDIT_ACTN type="CHAR"/>
      <BASE_LANGUAGE_CD type="CHAR"/>
      <MSG_SEQ_FLG type="CHAR"/>
      <PROCESS_INSTANCE type="NUMBER"/>
      <PUBLISH_RULE_ID type="CHAR"/>
      <MSGNODENAME type="CHAR"/>
    </PSCAMA>
  </FieldTypes>
  </xsl:template>
```

The interesting thing about this template is that it is basically an instruction to insert static text into the output document. Of course, this makes sense, as the FieldTypes section is dependant upon the message structure, and not the actual data contained within any particular message instance. Also note that there is no further node selection in this template, so after the XML is emitted this particular path through the XML ends.

3. Now that the FieldTypes nodes have been processed, the transform engine processes the MsgData node using the following matching template:

```
<xsl:template match="MsgData">
    <MsgData>
      <xsl:apply-templates select="Transaction"/>
    </MsgData>
 </xsl:template>
```

The template is quite simple. The transformation engine is to put a starting MsgData node in the output document, and then process the Transaction nodes in the input document. Note that the node context has changed: the current node in the input document being processed is the MsgData node. The call to select then implies that all child Transaction nodes under the MsgData are to be selected.

4. The Transaction nodes under MsgData are matched by the following template:

```
<xsl:template match="Transaction">
    <Transaction>
      <xsl:apply-templates select="PURCHASEORDER"/>
      <PSCAMA class="R">
        <LANGUAGE_CD>ENG</LANGUAGE_CD>
        <AUDIT_ACTN/>
        <BASE_LANGUAGE_CD>ENG</BASE_LANGUAGE_CD>
        <MSG_SEQ_FLG/>
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
        <PROCESS_INSTANCE>0</PROCESS_INSTANCE>
        <PUBLISH_RULE_ID/>
        <MSGNODENAME/>
      </PSCAMA>
    </Transaction>
  </xsl:template>
```

A Transaction start tag is written to the output document, and then the PURCHASEORDER nodes are to be handled. Once these nodes have been processed, the PSCAMA information will be written out, along with the closing Transaction tag.

5. The call to handle the PURCHASEORDER node invokes:

```
<xsl:template match="PURCHASEORDER">
   <PO_HEADER class="R">
     <PO_NUMBER IsChanged="Y">
        <xsl:value-of select="PURCHASEORDERNUM"/>
     </PO_NUMBER>
     <PO_DATE IsChanged="Y">
        <xsl:value-of select="PURCHASEORDERDATE"/>
     </PO_DATE>
     <xsl:apply-templates select="PURCHASEDITEMS"/>
   </PO_HEADER>
 </xsl:template>
```

The PO_HEADER start tag is emitted as well as the child PO_NUMBER and PO_DATE elements. The call out to xs:value-of means that node values from the input message are copied to the output message. In this case, the node PO_NUMBER in the output message is given the value from PURCHASEORDERNUM in the input message. PO_DATE is given the value from PURCHASEORDERDATE. Once these two elements have been written out, the transformation engine is then told to process the PURCHASEDITEMS nodes in the input document.

6. Each PURCHASEDITEMS node in the input message causes the following template to be executed:

```
<xsl:template match="PURCHASEDITEMS">
   <PO_ITEM class="R">
     <SKU IsChanged="Y"><xsl:value-of select="ITEM"/></SKU>
     <CUSTNAME IsChanged="Y">
        <xsl:value-of select="../SHIPPINGDETAILS/NAME"/>
     </CUSTNAME>
     <SHIPPER IsChanged="Y">
        <psft_function name="codeset" codesetname="SCM_CODESET">
           <parm name="CARRIER_ID">
             <xsl:value-of select="../SHIPPINGDETAILS/CARRIER_ID"/>
           </parm>
           <value name="SHIPPER" select="."/>
        </psft_function>
      </SHIPPER>
     <DESTADD IsChanged="Y">
        <xsl:value-of select="../SHIPPINGDETAILS/ADDRESS"/>
     </DESTADD>
     <DESTCITY IsChanged="Y">
        <xsl:value-of select="../SHIPPINGDETAILS/CITY"/>
```

Copyright © 1988-2007, Oracle. All rights reserved.

```
    </DESTCITY>
    <DESTSTATE IsChanged="Y">
       <xsl:value-of select="../SHIPPINGDETAILS/STATE"/>
     </DESTSTATE>
  </PO_ITEM>
</xsl:template>
```

This is where the bulk of the building of the output message is performed. For each PURCHASEDITEMS node in the input document, this template will be run once. The template is responsible for building out the PO_ITEM element and children in the output message. As in the template for PURCHASEORDER, this template uses the values from the input message and copies them across to the output message. For example, the SKU element in the output is given the value from the ITEM node in the input. Also note that the SHIPPER node contains a reference to psft-function and the SCM_CODESET. At this stage in the transformation, this text is static except for the value-of call to "../SHIPPINGDETAILS/CARRIER_ID", which will be resolved to "USPS". The actual codeset lookup will not be done at this point; this will occur in the second pass.

7. After the PURCHASEDITEMS template completes, the transformation jumps back to step 5, and outputs the closing PO_HEADER template.
8. Once the PURCHASEORDER template in step 5 completes, the transformation jumps back to step 4, and the Transaction template is completed. At this point the PSCAMA section in the template is written to the output message.
9. After the Transaction template in step 4 completes, the transformation jumps back to step 3, and the MsgData template. At this point the closing MsgData tag is written to the output message.
10. After the MsgData template in step 3 completes, the transformation jumps back to step 1. At this point the closing PO_MSG tag is written out, and the first part of the transformation process ends.

Transformation Processing: Second Pass

This section discusses the second pass of transform processing, during which the psft_function calls are resolved and the codeset values are placed into the document.

Assuming that no errors were encountered in the processing in the first pass a complete XML document will be available, containing the formatted PeopleSoft CRM purchase order message. However, this XML will still contain the reference:

```
<SHIPPER IsChanged="Y">
    <psft_function name="codeset" codesetname="SCM_CODESET">
       <parm name="CARRIER_ID">USPS</parm>
       <value name="SHIPPER" select="."/>
    </psft_function>
 </SHIPPER>
```

The second pass walks through the message and resolves all calls to psft_function. In this instance the codeset lookup will be run, and the psft_function node will be replaced with the result. The XML in the output message will then look like:

```
<SHIPPER IsChanged="Y">
    United States Postal Service
 </SHIPPER>
```

After the second pass completes, the transform is complete.

 Copyright © 1988-2007, Oracle. All rights reserved.

Testing the Transformation

The Transform Test utility is very useful when creating message transforms, since it allows one to exercise the transform without actually having service operations being invoked.

This section discusses using the utility to verify the example in this appendix.

To test the transformation:

1. Save to file the example message XML from the Message Definition: PeopleSoft SCM Node section presented earlier in this appendix.
2. Put the XSL from the Setting Up The Transformation section of this appendix in an application engine program.
3. Access the Transformation Test utility.

   Select PeopleTools, Integration Broker, Service Utilities, Transformation Test.

   The Transformation Test Utility page appears.
4. Add a new project.
5. In the *Program Name* field, enter in the name of the application engine program containing the XSL.
6. In the *Source Node Name* field, enter in the node name for the PeopleSoft SCM node.
7. In the *Destination Node Name* field enter in the node name for the CRM node.
8. In the *File Name* field, enter in the fully qualified path name of the file containing the input message.
9. Click the *Transform* button.

   The transformed XML appears in the Message Text text box.

Copyright © 1988-2007, Oracle. All rights reserved.

 Copyright © 1988-2007, Oracle. All rights reserved.

APPENDIX D

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.

Copyright © 1988-2007, Oracle. All rights reserved.

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	<PS_HOME>\webserv\<DOMAIN>\applications\peoplesoft\PSIGW\SDK
Java classes	<PS_HOME>\webserv\<DOMAIN>\applications\peoplesoft\PSIGW\WEB-INF\classes
API documentation	<PS_HOME>\webserv\<DOMAIN>\applications\peoplesoft\PSIGW\SDK\docs \index.html
Sample code for listening and target connector classes	<PS_HOME>\webserv\<DOMAIN>\applications\peoplesoft\PSIGW\SDK\src
Send Master utility	<PS_HOME>\webserv\<DOMAIN>\StartSendMaster.bat, or <PS_HOME>\webserv \<DOMAIN> \StartSendMaster.sh
Simple Post utility	<PS_HOME>\webserv\<DOMAIN>\applications\peoplesoft\PSIGW\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, <PS_HOME>\webserv \<DOMAIN>\applications\peoplesoft\PSIGW\SDK\docs\index.html.

Copyright © 1988-2007, Oracle. All rights reserved.

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).

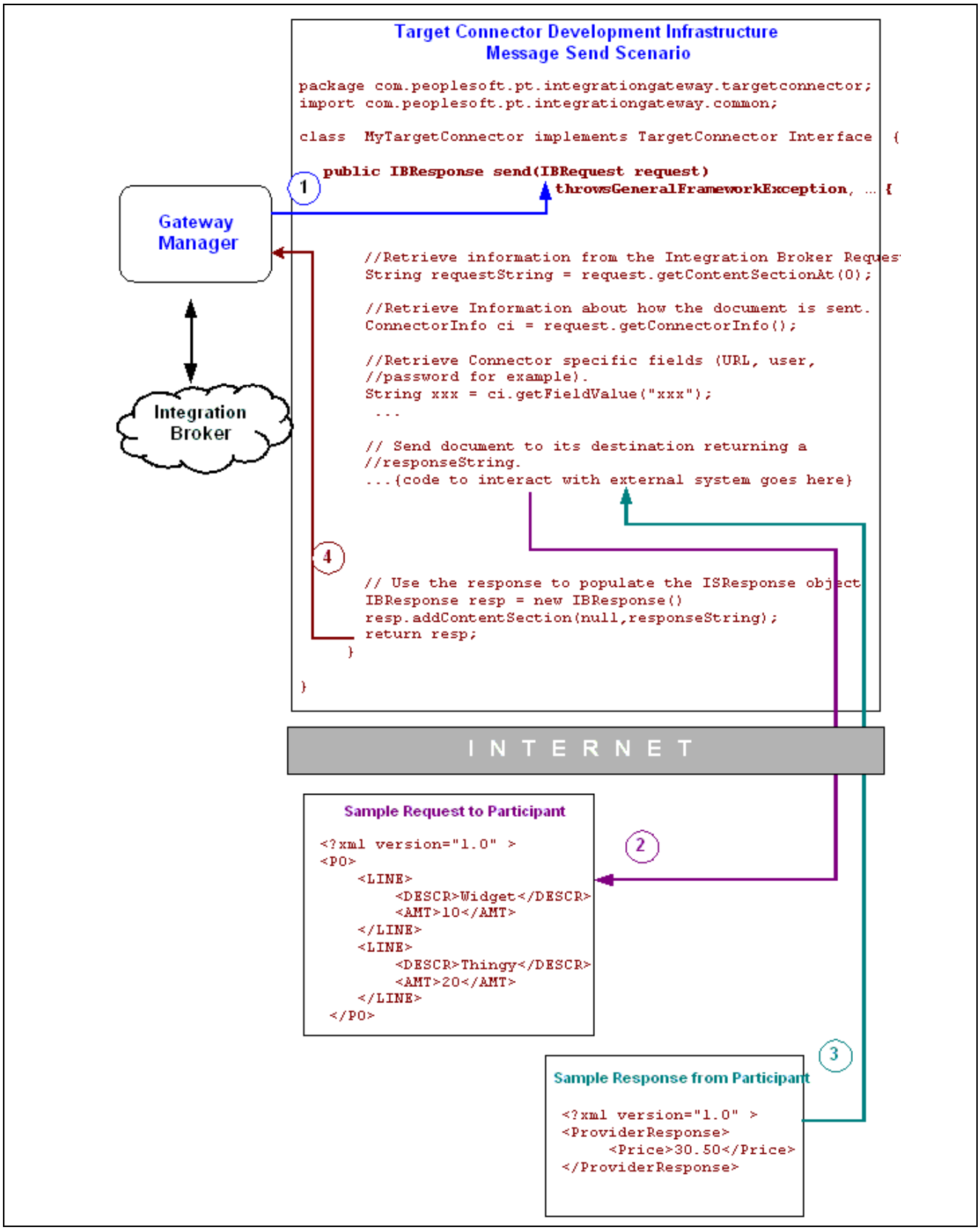
Target and listening connector templates are provided at the end of this chapter for you to use as a starting point for connector development.

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:

Copyright © 1988-2007, Oracle. All rights reserved.

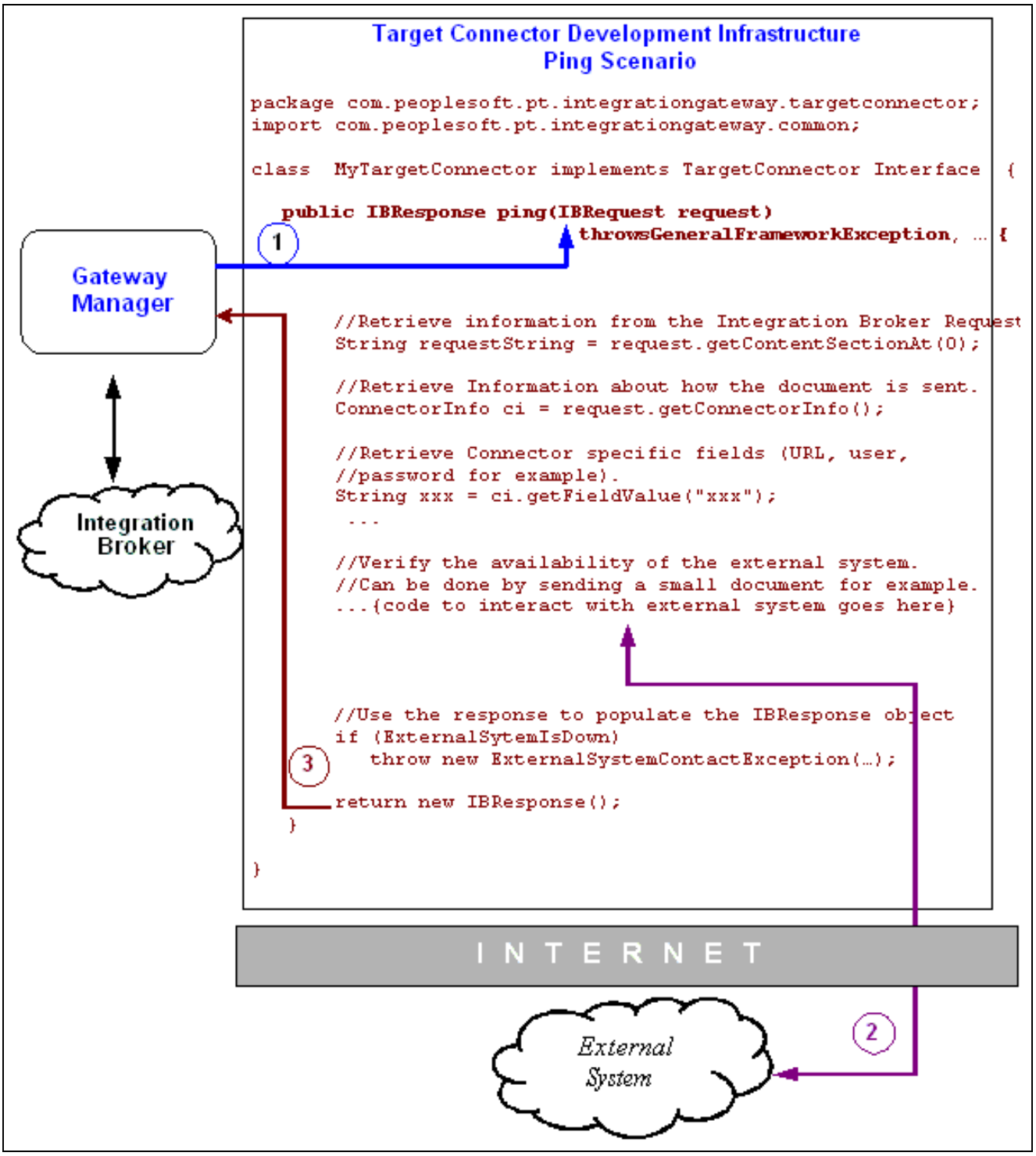


Message send scenario

Copyright © 1988-2007, Oracle. All rights reserved.

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:

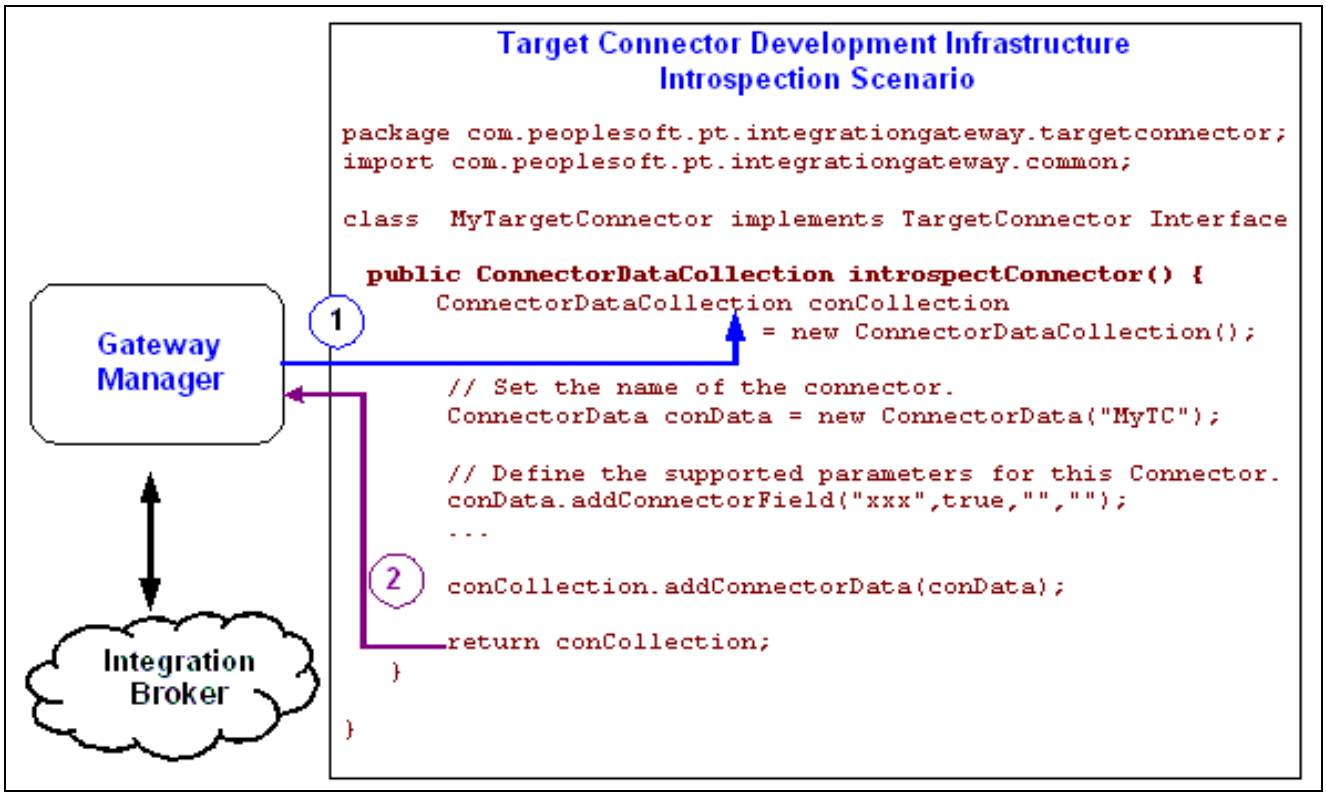


Ping scenario

Copyright © 1988-2007, Oracle. All rights reserved.

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.

Copyright © 1988-2007, Oracle. All rights reserved.

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.

Copyright © 1988-2007, Oracle. All rights reserved.

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:

 Copyright © 1988-2007, Oracle. All rights reserved.

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.
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,
```

Copyright © 1988-2007, Oracle. All rights reserved.

```
                    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");
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
   } 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 D, "Using the Integration Broker Connector SDK," Using Connector Templates, page 766.

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.

Copyright © 1988-2007, Oracle. All rights reserved.

- 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=8.49

#
## JOLT connect string settings for Application Server(s) with known NODENAMEs.
#
ig.isc.FINANCIALS.serverURL=//FINSERVER:9000
ig.isc.FINANCIALS.userid=FINUSERID
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
ig.isc.FINANCIALS.password=FINPASSWORD
ig.isc.FINANCIALS.toolsRel=8.49

ig.isc.CRM.serverURL=//CRMSERVER:9000
ig.isc.CRM.userid=CRMUSERID
ig.isc.CRM.password=CRMPASSWORD
ig.isc.CRM.toolsRel=8.49
```

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 {
```

Copyright © 1988-2007, Oracle. All rights reserved.

```
            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();
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
            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:

<PS_HOME>\webserv\<DOMAIN>\applications\peoplesoft\PSIGW\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:

<PS_HOME>\webserv\<DOMAIN>\applications\peoplesoft\PSIGW\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.

Copyright © 1988-2007, Oracle. All rights reserved.

See Also

Chapter 7, "Managing Integration Gateways," Loading Target Connectors, page 57

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;
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
    }

    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;
```

Copyright © 1988-2007, Oracle. All rights reserved.

```
//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:
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
        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>
```

Copyright © 1988-2007, Oracle. All rights reserved.

```
                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.

 Copyright © 1988-2007, Oracle. All rights reserved.

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.
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();
```

Copyright © 1988-2007, Oracle. All rights reserved.

```
   } 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>";
}
```


Copyright © 1988-2007, Oracle. All rights reserved.

See Also

Enterprise PeopleTools 8.49 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.

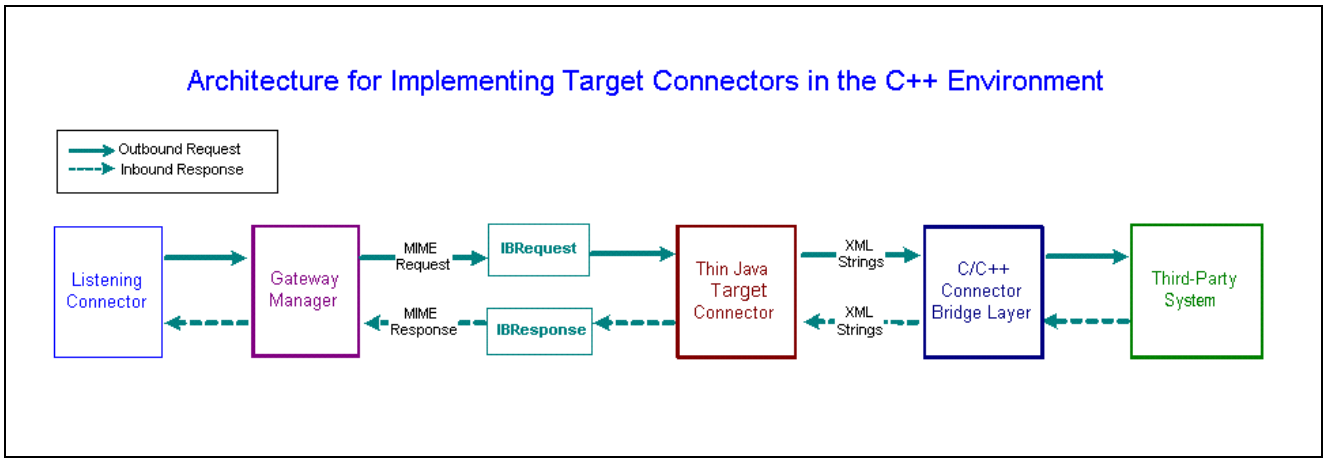
Java Target Connectors

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:

Copyright © 1988-2007, Oracle. All rights reserved.



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
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
        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):
```

Copyright © 1988-2007, Oracle. All rights reserved.

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 BEA 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

Appendix D, “Using the Integration Broker Connector SDK,” Installing Connector Classes, page 765

Appendix D, “Using the Integration Broker Connector SDK,” Registering Connectors, page 765

Chapter 7, “Managing Integration Gateways,” Loading Target Connectors, page 57

 Copyright © 1988-2007, Oracle. All rights reserved.

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
```

Copyright © 1988-2007, Oracle. All rights reserved.

```
                        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;
   }
}
```

 Copyright © 1988-2007, Oracle. All rights reserved.

APPENDIX E

Understanding Migrated Integration Metadata

This appendix discusses:

- Migrated integration metadata.
- Migrated integration PeopleCode.
- Correcting integration PeopleCode that did not successfully migrate.

Understanding Migrated Integration Metadata

The following table summarizes how objects are migrated between PeopleTools 8.47 and earlier releases and PeopleTools 8.48 and higher releases:

PeopleTools 8.4x Objects (PeopleTools 8.47 and Earlier)	PeopleTools 8.48 and Higher Objects
Node.	Node
Channel.	Queue.
Message.	Message.
Node transactions and relationships.	Service, Service Operations, Service Operation Versions, and Routings.
Message and Subscription PeopleCode.	Application classes and service operation handlers.

Note. All objects migrated to PeopleTools 8.48 and higher releases have the Owner ID of the message from which they were created. Any invalid Owner IDs are deleted at the time of upgrade. If no Owner ID exists for an object, you must manually define one in the PeopleTools 8.48 or higher database.

Node Objects

Upgraded node objects are assigned a Default User ID as defined in the upgrade script.

In PeopleTools 8.48 and higher releases security is implemented at the user ID level. The default user ID is used in conjunction with securing service operations.

Channel Objects

Channels have been converted to queues.

Copyright © 1988-2007, Oracle. All rights reserved.

The new queue name should be the same as the old channel name

As part of the upgrade/conversion, any related language data associated with the channel should also be brought over to the newly created queue definition.

Message Objects

Messages are shapes that provide the physical description of the contents of a service operation transaction, and describe the data being sent, including fields, field types, and field lengths.

Unlike PeopleTools 8.47 and earlier releases, beginning with PeopleTools 8.48 messages do not contain any processing logic, such as PeopleCode events or subscriptions. All processing logic is created by extending a set of delivered application classes, and associating those application classes to service operations using service operation handlers.

Node Transaction and Relationship Objects

Node transactions and relationships are migrated to services, service operations and routing definitions.

Service Objects

During the upgrade process, a service definition is created for each distinct message referenced in the node transaction table.

A service inherit most of its properties from the message, including description, long description, Owner ID, language code, and so on.

The service name in PeopleTools 8.48 and higher releases is the same as the message name in the PeopleTools 8.47 or earlier system.

Any related language data associated with the message is inherited by the service.

Service Operation Objects

Service operations have the same name as the service to which they are associated.

Complex transactions like asynchronous or synchronous transactions with transformations, hub transactions or async-to-sync are defined by grouping a set of node transactions and relationships together. As the complex cases are upgraded, the system separates node transactions that were created for these complex cases (for example an asynchronous hub case) from the simple cases (for example, outbound asynchronous requests with no transformation). Asynchronous and synchronous node transactions that do not participate in a relationship (for example, those that are left over after we settled the complex cases) in PeopleTools 8.47 and earlier releases become service operations in PeopleTools 8.48 and higher releases.

Warning! If there is no node transaction on the PeopleTools 8.47 or earlier system, no service operation is created on the PeopleTools 8.48 or higher system during upgrade.

In cases where a PeopleTools 8.47 or earlier node has both synchronous and asynchronous transactions, the first transaction defined on the node is migrated as a service operation; the second message cannot be created and an error is generated to the output log at the time of upgrade.

For asynchronous-to-synchronous transactions on a PeopleTools 8.47 or earlier system, the following occurs during the upgrade process to PeopleTools 8.48 or higher system:

- The service operation is named after the outbound message name.
- The request message is named after the outbound asynchronous message.

 Copyright © 1988-2007, Oracle. All rights reserved.

- The response message is named after the asynchronous response message specified in the relationship.

In PeopleTools 8.47 and earlier systems IsActive was used to check the active property on the message, in PeopleTools 8.48 and higher systems it checks the service operation. In cases where a service operation is not created for a message, IsActive returns *False*. Therefore there may be a behavioral change from previous releases. In cases where there is no service operation created, and you want the previous behavior preserved, you must create a service operation and set the operation state to match that which was on the message.

Routing Objects

The upgrade process creates point-to-point routing definitions in the PeopleTools 8.48 or higher system based on node transactions and relationships defined in the PeopleTools 8.47 or earlier system.

Only current relationships in the PeopleTools 8.47 or earlier system are migrated.

The Active/Inactive flag on the transaction in the PeopleTools 8.47 or earlier system determines whether the routing definition is active or inactive in the PeopleTools 8.48 or later system.

The connector information defined on the outbound transaction is used in the routing definition. You must manually update aliases on routing definitions that use custom connectors.

When the system creates routing definitions during the upgrade process, the external message name from the transaction is used as the routing alias, if one was defined. If there is no value in the external message name, messages on nodes that are marked as *External* use the PeopleTools 8.47 or earlier system alias message name. PeopleTools 8.47 and earlier system nodes marked as anything other then *External* use the PeopleTools 8.48 alias format of *serviceoperation.version*.

All routing definitions created during the upgrade process are point-to-point routings, with one exception. In cases where on the PeopleTools 8.47 or earlier system there is an inbound synchronous or asynchronous node transaction on the default local node without a corresponding outbound synchronous or asynchronous node transaction (via relationship) on the default local node, an any-to-local routing definition is created during the upgrade.

Understanding Migrated Integration PeopleCode

The following table summarized how PeopleCode has been migrated between PeopleTools 8.4x releases and PeopleTools 8.48:

PeopleTools 8.4x PeopleCode (PeopleTools 8.47 and Earlier)	PeopleTools 8.48 and Higher PeopleCode
Message and subscription PeopleCode.	Application classes.
PeopleCode events.	Service operation handlers.

All PeopleCode referenced in a message is converted to an application class, which is in turn then referenced by a handler that is created in the service operation generated by the converted message.

Application Classes

PeopleCode defined on messages in PeopleTools 8.47 and earlier releases is migrated into application classes in PeopleTools 8.48 and higher releases.

Copyright © 1988-2007, Oracle. All rights reserved.

Application classes have to belong to an application package. The message name becomes the application package name and description. The exception for this naming convention is when an application package already exists on the PeopleTools 8.48 or higher system that matches the message name. In this case the application package is stripped of all underscores and the number *1* is appended to the end of the name.

For every subscription event associated with a message, an application class is created in the PeopleTools 8.48 or higher system with a similar name as the subscription event. Since application class names cannot contain special characters, those found in the subscription event name are simply replaced with an underscore.

Default values for application class names are used for the other message events.

If a PeopleTools 8.47 or earlier message has no PeopleCode defined on it, no application package is generated for it in the PeopleTools 8.48 or higher system.

Application classes that fail to compile are saved and commented out. In these cases, a deprecated handler is created to invoke the old message or subscription event, to behave, runtime-wise, as it did in PeopleTools 8.47 and earlier systems. The user, however, is responsible for correcting any application classes that failed to compile and modifying any service operation definition that makes use of the deprecated handler.

See Appendix E, "Understanding Migrated Integration Metadata," Correcting Integration PeopleCode That Did Not Migrate, page 783.

PeopleCode Methods

The following table describes how PeopleCode events from PeopleTools 8.47 and earlier releases map to PeopleTools 8.48 and higher methods.

PeopleTools 8.4x Integration PeopleCode Events (PeopleTools 8.47 and Earlier)	PeopleTools 8.48 Integration PeopleCode Base Classes and Methods (PeopleTools 8.48 and Higher)
OnRequest	IRequestHandler:OnRequest
OnAckReceive	IReceiver:OnAckReceive
OnRouteReceive	IRouter: OnRouteReceive
OnRouteSend	IRouter:OnRouteSend
OnSend	ISend:OnRequestSend
Subscription	INotification:OnNotify

Built-In Functions

Many PeopleCode built-in functions have been deprecated for the new PeopleSoft Integration Broker model. Most of the PeopleTools 8.47 and earlier built-in functions have been internally redirected to work with the PeopleTools 8.48 and later equivalent. When compiling PeopleCode that uses the PeopleTools 8.47 and earlier built-in functions, an informational message appears that explains that the given built-in has been deprecated.

Other Migrated Constructs

The following constructs are also migrated during the upgrade process:

- Import statements.

 Copyright © 1988-2007, Oracle. All rights reserved.

- Function libraries.
- Variables not explicitly declared.

Special Characters

During the upgrade process, any special characters that used in the name of PeopleCode constructs in the PeopleTools 8.47 or earlier PeopleCode, such as slashes (“/”), single quotation marks “ ’ ’ “, and periods (.) are replaced with underscores (_).

Correcting Integration PeopleCode That Did Not Migrate

This section provides an overview of integration PeopleCode that did not migrate and how to correct it.

Understanding Integration PeopleCode That Did Not Migrate

This section discusses the reasons why integration events and subscriptions might not successfully migrate to application classes during the upgrade from PeopleTools 8.47 and earlier systems to PeopleTools 8.48 and higher systems and the *Deprecated* type service operation handler used to manage such PeopleCode.

Reasons Why Integration PeopleCode Does Not Migrate

Integration events and subscriptions might not successfully migrate to application classes for the following reasons:

- Use of global variables.
- Use of component variables.
- Use of constants.
- Use of local functions.
- Inability to determine the return type or the return type is incorrect.

Deprecated PeopleCode Handler

The PeopleSoft system creates a deprecated PeopleCode handler for any integration PeopleCode that cannot be migrated to the PeopleTools 8.48 or higher system.

Deprecated handlers enable you to run PeopleTools 8.47 and earlier PeopleCode (subscription or onRequest). However, PeopleSoft recommends that you correct the PeopleCode and migrate the code into an *Application Class* type handler for use in the PeopleTools 8.48 or higher system.

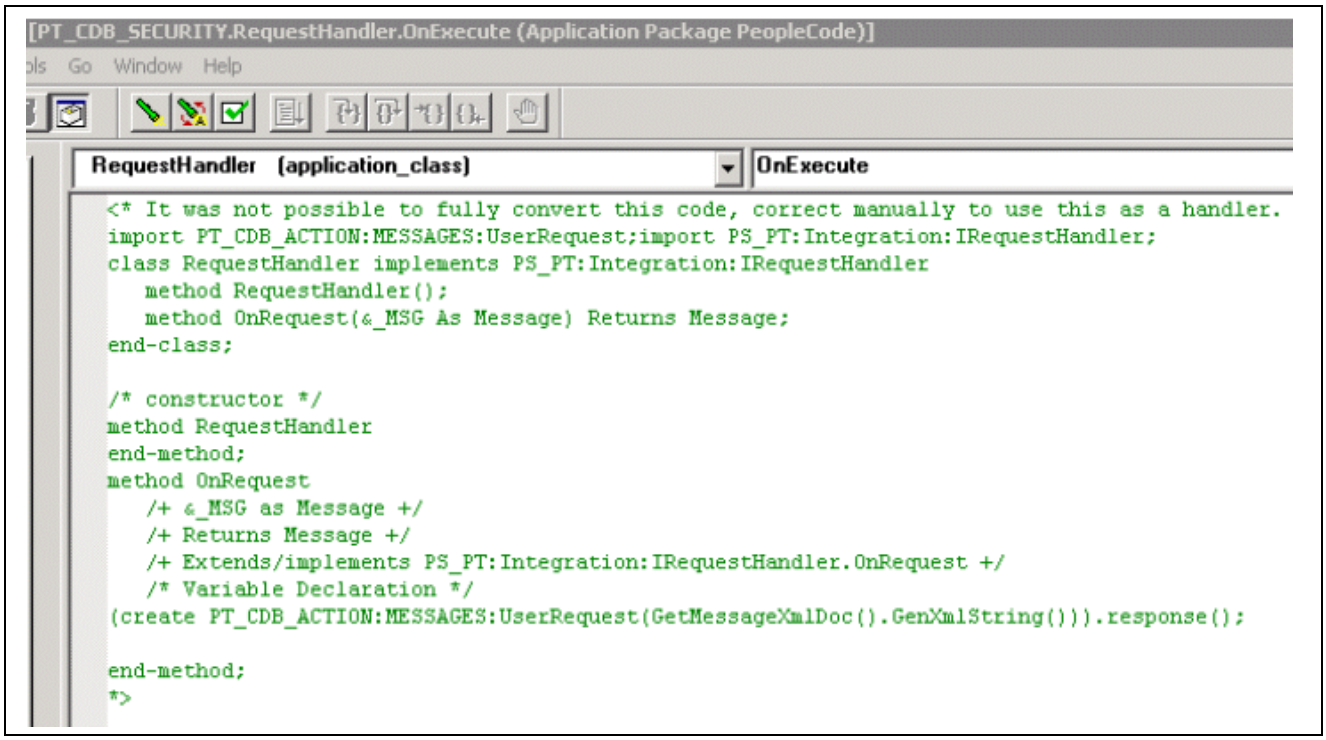
See Appendix E, “Understanding Migrated Integration Metadata,” Correcting Non-Migrated Integration PeopleCode, page 783.

Correcting Non-Migrated Integration PeopleCode

This section discusses correcting non-migrated PeopleCode and creating an *Application Class* type handler.

The following example shows an application class created for the PeopleTools 8.47 or earlier message *PT_CDB_SECURITY*. During the upgrade process, the system was unable to migrate the integration PeopleCode defined on the message.

Copyright © 1988-2007, Oracle. All rights reserved.



A PeopleTools 8.48 application class created for the PeopleTools 8.4x message PT_CDB_SECURITY

To correct non-migrated integration PeopleCode:

1. In PeopleSoft Application Designer, open the migrated application package.

   The package name is generally the same as the message name in the PeopleTools 8.47 or earlier system.
2. Remove the "<*" and "*>" symbols, as well as the comments from the code.
3. Compile the code.

   If the code did not migrate for one of the following reasons, the system displays a meaningful error message to assist you in correcting the problem:

   - Use of component scope variables.
   - Use of global variables or constant variables
   - Incorrect return type for handler.

   If the reason for the failure is due to the use of a local function, modify the PeopleCode to ensure that the function is passing appropriate types for the context in the application package.
4. Correct the PeopleCode and save the changes.
5. Delete the deprecated handler.

   a. Select PeopleTools, Integration Broker, Service Utilities, Service Administration.. Click the Deprecated PeopleCode tab.

      The Deprecated PeopleCode page appears.

   b. In the Service Operation field, enter the name of the service operation that contains the deprecated handler to delete and click the Search button.

   c. In the Select column, check the box next to the service operation name that contains the deprecated handler to delete.

   d. Click the Delete button.

Copyright © 1988-2007, Oracle. All rights reserved.

6. Add an *Application Class* type handler to the service operation definition that references the modified PeopleCode.

   See Chapter 15, "Managing Service Operations," Adding Handlers to Service Operations, page 301.

7. In the message definition that was migrated to the PeopleTools 8.48 or higher system, delete the PeopleCode event or Subscription that is defined on the message.

   Open the PeopleCode editor for the respective event and null out the program that exists there. Several warnings appear when saving the program, but your changes will be committed.

Copyright © 1988-2007, Oracle. All rights reserved.

Copyright © 1988-2007, Oracle. All rights reserved.

APPENDIX F

Data Dependencies and Relationships When Copying Data Between Databases

This appendix discusses:

- Data dependencies and relationships for copying data.
- Using data mover scripts to move non-managed object data.

Understanding Data Dependencies and Relationships for Copying Data

When copying PeopleSoft Integration Broker data between PeopleTools 8.49 databases, you must be aware of data dependencies and relationships to ensure that no errors occur and to lessen the chance of encountering orphaned data.

You can use the PeopleSoft Application Designer Project Copy feature to copy managed object data between PeopleTools 8.49 databases. PeopleSoft provides data mover scripts for moving non-managed object data.

Note. References to 'Project Copy' in the following table are references to the Project Copy feature in PeopleSoft Application Designer.

Object Name	Managed Object	Comments
Services.	Yes.	You can use Project Copy to copy services between databases. WSDL documents that exist for a service are not automatically copied with a service. Since WSDL documents are not managed objects you cannot include them in your copy project. However, PeopleSoft provides several data mover scripts to export and import WSDL documents into PeopleTools 8.49 databases.
WSDL documents.	No.	WSDL documents are not managed objects, so you cannot use Project Copy to copy WSDL to a different database. PeopleSoft provides data mover scripts for importing and exporting WSDL between PeopleTools 8.49 databases.

Copyright © 1988-2007, Oracle. All rights reserved.

Object Name	Managed Object	Comments
Service operations.	Yes.	A service operation is tied to a service. If you copy service operation in a project, the target database must already contain the service to which the service operation is tied in the database. If it does not, you must include that service in the copy project. Service operations cannot exist in a database without at least one service operation version - the default version. So when copying a service operation between databases, you need to be aware what the default service operation is and that you may possible have to copy it to the target database as well.
Service operation versions.	Yes.	A service operation version refers to a specific service operation. If you copy a service operation version, the target database must already contain the service operation. If it does not, you must include that service operation in the copy project. In addition, service operation versions refer to messages. If you copy a service operation version, the messages that are referenced for that service operation version, must exist on the target database. If they do not, you must include them in the copy project. If WSDL documents have been generated for a service operation version, they are not automatically copied during the Project Copy process. Further, once you have copied a service operation version to the target database, it may appear that WSDL documents exist for a service operation version, when they do not. To avoid this situation, after you copy a service operation version to the target database, open the service definition to which the service operation belongs. If the View WSDL link appears, and when you click it WSDL appears, go back to the source database and export the generated WSDL documents to the target database. Another option is to delete the WSDL documents associated with a service operation before the Project Copy, and regenerate them on the target database.
Service operation handlers.	Yes.	A service operation handler refers to a specific service operation. If you copy a service operation handler, the target database must already contain the service operation to which the handler refers. If it does not, you must include that service operation in the copy project.

Copyright © 1988-2007, Oracle. All rights reserved.

Object Name	Managed Object	Comments
Service operation routings.	Yes.	Routing names are keys in the system. If you copy a routing, the sending and receiving nodes must defined on the target database. If they are not defined on the target database, you must include them in your copy project. Routings reference a specific service operation version. If you copy a routing, the target database must already contain the service operation version to which the routing refers. If it does not, you must include that service operation version in the project. Routings also reference nodes. If you copy a routing, the target database must already contain the nodes being referenced. An exception to this is the local default node. During project copy, any routing referencing the local default node will be modified to reference the default local node of the target system.
Messages.	Yes.	Container messages and message parts must have message schemas to function properly. PeopleSoft provides several data mover scripts for exporting and importing message schemas between databases.
IB queues.	Yes.	NA
Message schemas.	No.	Message schemas are not managed objects, so you cannot use Project Copy to copy schemas to a different database. PeopleSoft provides data mover scripts for importing and exporting schema between PeopleTools 8.49 databases.

See Also

Enterprise PeopleTools 8.49 PeopleBook: PeopleSoft Application Designer, "Copying Projects and Definitions"

Using Data Mover Scripts to Move Non-Managed Object Data

The following table lists the data mover scripts that PeopleSoft provides to move WSDL documents and message schemas between PeopleTools 8.49 databases. These scripts are located in the <PS_HOME>\scripts directory.

Object	Script Name	Description
Message schema.	PSIBMSGSCHEMA_EXP.DMS	Export a message schema from a PeopleTools 8.49 database.
Message schema.	PSIBMSGSCHEMA_IMP.DMS	Import a message schema into a PeopleTools 8.49 database.

Copyright © 1988-2007, Oracle. All rights reserved.

Object	Script Name	Description
WSDL document.	PSIBWSDL_EXP.DMS	Export a WSDL document from a PeopleTools 8.49 database.
WSDL document.	PSIBWSDL_IMP.DMS	Import a WSDL document into a PeopleTools 8.49 database.

The WSDL data mover scripts move WSDL by WSDL name, not service name. Therefore it is possible to select specific WSDL for importing/exporting for a given service.

You may encounter errors while moving large WSDL documents and schemas from a Microsoft SQL/Oracle platform to a DB2/Sybase platform, because of size restrictions in Sybase and DB2. The maximum size on DB2 databases is 16349. The maximum size on Sybase databases is 31999 bytes.

 Copyright © 1988-2007, Oracle. All rights reserved.

APPENDIX G

Technologies, Specifications, and Products Supported by PeopleSoft Integration Broker

This appendix discusses technologies, specifications and third-party products that are used in or are supported by PeopleSoft Integration Broker.

Supported technologies, specifications and Third-Party Products

The following table lists the technologies, specifications and third-party products that are used in or are supported by PeopleSoft Integration Broker.

Technology/Product	Version/Specification	Comments
Apache Xalan	1.9. C++ version.	Apache Xalan is an XSLT processor for transforming XML documents into HTML, text, or other document type.
Apache Xerces	2.6.0. C++ version	Apache Xerces is an XML parser.
AS2	RFC 3335	Based on software product IP*Works! EDI V6.
FTP/FTPS	5.1	NA
HTTP	1.1	NA
HTTPS	SSL 3.0	NA
iWay SOAPswitch	5.5.3	NA
JMS	1.0.2	NA
Oracle XSL Mapper	10.1.2.02	NA
SOAP	1.1 and 1.2	All SOAP output from PeopleSoft Integration Broker is version 1.1. PeopleSoft Integration Broker understands versions 1.1 and 1.2. Uses Schema for the SOAP/1.1 envelope, SOAP/1.2 http://schemas.xmlsoap.org/soap/envelope/.
UDDI	2.0	NA

Copyright © 1988-2007, Oracle. All rights reserved.

Technology/Product	Version/Specification	Comments
WSDL	1.1	Schema used is http://schemas.xmlsoap.org/wsdl.
WS-Addressing	1.1	Currently the WS-Addressing document is a formal submission to W3C, the current submission is using schema under http://schemas.xmlsoap.org/ws/2004/08/addressing/, however, the integration gateway uses an older version, http://schemas.xmlsoap.org/ws/2003/03/addressing/.
WS-I Basic Profile	1.0	The WS-I Basic Profile 1.0 specification mandates support for SOAP 1.1, WSDL 1.1, UDDI 2.0, HTTP 1.1, SSL 3.0 (or HTTPS), XML 1.0 (2nd Edition), and XML Schema (Part 1 and 2).
WS-Interoperability	1.2.1	NA
XML	1.0 (Second edition.)	NA
XML schema	Part 1 and 2	NA
XSLT	1.0	By use of Apache Xerces 2.6.0 and Apache Xalan 1.9, XSLT 1.0 is supported.
XQuery, XPath	1.0	Used in PeopleSoft Integration Broker transform programs. By use of Apache Xerces 2.6.0 and Apache Xalan 1.9, XQuery and XPath 1.0 is supported.

Copyright © 1988-2007, Oracle. All rights reserved.

Glossary of PeopleSoft Enterprise Terms

absence entitlement This element defines rules for granting paid time off for valid absences, such as sick time, vacation, and maternity leave. An absence entitlement element defines the entitlement amount, frequency, and entitlement period.

absence take This element defines the conditions that must be met before a payee is entitled to take paid time off.

academic career In PeopleSoft Enterprise Campus Solutions, all course work that a student undertakes at an academic institution and that is grouped in a single student record. For example, a university that has an undergraduate school, a graduate school, and various professional schools might define several academic careers—an undergraduate career, a graduate career, and separate careers for each professional school (law school, medical school, dental school, and so on).

academic institution In PeopleSoft Enterprise Campus Solutions, an entity (such as a university or college) that is independent of other similar entities and that has its own set of rules and business processes.

academic organization In PeopleSoft Enterprise Campus Solutions, an entity that is part of the administrative structure within an academic institution. At the lowest level, an academic organization might be an academic department. At the highest level, an academic organization can represent a division.

academic plan In PeopleSoft Enterprise Campus Solutions, an area of study—such as a major, minor, or specialization—that exists within an academic program or academic career.

academic program In PeopleSoft Enterprise Campus Solutions, the entity to which a student applies and is admitted and from which the student graduates.

accounting class In PeopleSoft Enterprise Performance Management, the accounting class defines how a resource is treated for generally accepted accounting practices. The Inventory class indicates whether a resource becomes part of a balance sheet account, such as inventory or fixed assets, while the Non-inventory class indicates that the resource is treated as an expense of the period during which it occurs.

accounting date The accounting date indicates when a transaction is recognized, as opposed to the date the transaction actually occurred. The accounting date and transaction date can be the same. The accounting date determines the period in the general ledger to which the transaction is to be posted. You can only select an accounting date that falls within an open period in the ledger to which you are posting. The accounting date for an item is normally the invoice date.

accounting split The accounting split method indicates how expenses are allocated or divided among one or more sets of accounting ChartFields.

accumulator You use an accumulator to store cumulative values of defined items as they are processed. You can accumulate a single value over time or multiple values over time. For example, an accumulator could consist of all voluntary deductions, or all company deductions, enabling you to accumulate amounts. It allows total flexibility for time periods and values accumulated.

action reason The reason an employee's job or employment information is updated. The action reason is entered in two parts: a personnel action, such as a promotion, termination, or change from one pay group to another—and a reason for that action. Action reasons are used by PeopleSoft Enterprise Human Resources, PeopleSoft Enterprise Benefits

Copyright © 1988-2007, Oracle. All rights reserved.

Administration, PeopleSoft Enterprise Stock Administration, and the COBRA Administration feature of the Base Benefits business process.

action template In PeopleSoft Enterprise Receivables, outlines a set of escalating actions that the system or user performs based on the period of time that a customer or item has been in an action plan for a specific condition.

activity In PeopleSoft Enterprise Learning Management, an instance of a catalog item (sometimes called a class) that is available for enrollment. The activity defines such things as the costs that are associated with the offering, enrollment limits and deadlines, and waitlisting capacities.

In PeopleSoft Enterprise Performance Management, the work of an organization and the aggregation of actions that are used for activity-based costing.

In PeopleSoft Enterprise Project Costing, the unit of work that provides a further breakdown of projects—usually into specific tasks.

In PeopleSoft Workflow, a specific transaction that you might need to perform in a business process. Because it consists of the steps that are used to perform a transaction, it is also known as a step map.

address usage In PeopleSoft Enterprise Campus Solutions, a grouping of address types defining the order in which the address types are used. For example, you might define an address usage code to process addresses in the following order: billing address, dormitory address, home address, and then work address.

adjustment calendar In PeopleSoft Enterprise Campus Solutions, the adjustment calendar controls how a particular charge is adjusted on a student's account when the student drops classes or withdraws from a term. The charge adjustment is based on how much time has elapsed from a predetermined date, and it is determined as a percentage of the original charge amount.

administrative function In PeopleSoft Enterprise Campus Solutions, a particular functional area that processes checklists, communication, and comments. The administrative function identifies which variable data is added to a person's checklist or communication record when a specific checklist code, communication category, or comment is assigned to the student. This key data enables you to trace that checklist, communication, or comment back to a specific processing event in a functional area.

admit type In PeopleSoft Enterprise Campus Solutions, a designation used to distinguish first-year applications from transfer applications.

agreement In PeopleSoft Enterprise eSettlements, provides a way to group and specify processing options, such as payment terms, pay from a bank, and notifications by a buyer and supplier location combination.

allocation rule In PeopleSoft Enterprise Incentive Management, an expression within compensation plans that enables the system to assign transactions to nodes and participants. During transaction allocation, the allocation engine traverses the compensation structure from the current node to the root node, checking each node for plans that contain allocation rules.

alternate account A feature in PeopleSoft Enterprise General Ledger that enables you to create a statutory chart of accounts and enter statutory account transactions at the detail transaction level, as required for recording and reporting by some national governments.

analysis database In PeopleSoft Enterprise Campus Solutions, database tables that store large amounts of student information that may not appear in standard report formats. The analysis database tables contain keys for all objects in a report that an application program can use to reference other student-record objects that are not contained in the printed report. For instance, the analysis database contains data on courses that are considered

Copyright © 1988-2007, Oracle. All rights reserved.

	for satisfying a requirement but that are rejected. It also contains information on courses captured by global limits. An analysis database is used in PeopleSoft Enterprise Academic Advisement.
Application Messaging	PeopleSoft Application Messaging enables applications within the PeopleSoft Enterprise product family to communicate synchronously or asynchronously with other PeopleSoft Enterprise and third-party applications. An application message defines the records and fields to be published or subscribed to.
AR specialist	Abbreviation for *receivables specialist.* In PeopleSoft Enterprise Receivables, an individual in who tracks and resolves deductions and disputed items.
arbitration plan	The arbiter when multiple price rules match the transaction. This plan determines the order in which the price rules are applied to the transaction base price.
assessment rule	In PeopleSoft Enterprise Receivables, a user-defined rule that the system uses to evaluate the condition of a customer's account or of individual items to determine whether to generate a follow-up action.
asset class	An asset group used for reporting purposes. It can be used in conjunction with the asset category to refine asset classification.
attribute/value pair	In PeopleSoft Enterprise Directory Interface, relates the data that makes up an entry in the directory information tree.
auction event	In PeopleSoft Strategic Sourcing, a sourcing event where bidders actively compete against one another to achieve the best price or score.
audience	In PeopleSoft Enterprise Campus Solutions, a segment of the database that relates to an initiative, or a membership organization that is based on constituent attributes rather than a dues-paying structure. Examples of audiences include the Class of '65 and Undergraduate Arts & Sciences.
authentication server	A server that is set up to verify users of the system.
base time period	In PeopleSoft Enterprise Business Planning, the lowest level time period in a calendar.
benchmark job	In PeopleSoft Enterprise Workforce Analytics Solution, a benchmark job is a job code for which there is corresponding salary survey data from published, third-party sources.
bid response	In PeopleSoft Strategic Sourcing, the response by a bidder to an event.
billing career	In PeopleSoft Enterprise Campus Solutions, the one career under which other careers are grouped for billing purposes if a student is active simultaneously in multiple careers.
bio bit or bio brief	In PeopleSoft Enterprise Campus Solutions, a report that summarizes information stored in the system about a particular constituent. You can generate standard or specialized reports.
book	In PeopleSoft Enterprise Asset Management, used for storing financial and tax information, such as costs, depreciation attributes, and retirement information on assets.
branch	A tree node that rolls up to nodes above it in the hierarchy, as defined in PeopleSoft Tree Manager.
budgetary account only	An account used by the system only and not by users; this type of account does not accept transactions. You can only budget with this account. Formerly called "system-maintained account."

Copyright © 1988-2007, Oracle. All rights reserved.

budget check In commitment control, the processing of source transactions against control budget ledgers, to see if they pass, fail, or pass with a warning.

budget control In commitment control, budget control ensures that commitments and expenditures don't exceed budgets. It enables you to track transactions against corresponding budgets and terminate a document's cycle if the defined budget conditions are not met. For example, you can prevent a purchase order from being dispatched to a vendor if there are insufficient funds in the related budget to support it.

budget period The interval of time (such as 12 months or 4 quarters) into which a period is divided for budgetary and reporting purposes. The ChartField allows maximum flexibility to define operational accounting time periods without restriction to only one calendar.

business activity The name of a subset of a detailed business process. This might be a specific transaction, task, or action that you perform in a business process.

business event In PeopleSoft Enterprise Receivables, defines the processing characteristics for the Receivable Update process for a draft activity.

In PeopleSoft Enterprise Sales Incentive Management, an original business transaction or activity that may justify the creation of a PeopleSoft Enterprise Incentive Management event (a sale, for example).

business process A standard set of 17 business processes are defined and maintained by the PeopleSoft Enterprise product families and are supported by the Business Process Engineering group. An example of a business process is Order Fulfillment, which is a business process that manages sales orders and contracts, inventory, billing, and so forth.

See also *detailed business process.*

business unit constraints In PeopleSoft Strategic Sourcing, these constraints apply to a selected Strategic Sourcing business unit. Spend is tracked across all of the events within the selected Strategic Sourcing business unit.

business task The name of the specific function depicted in one of the business processes.

business unit A corporation or a subset of a corporation that is independent with regard to one or more operational or accounting functions.

buyer In PeopleSoft Enterprise eSettlements, an organization (or business unit, as opposed to an individual) that transacts with suppliers (vendors) within the system. A buyer creates payments for purchases that are made in the system.

buy event In PeopleSoft Strategic Sourcing, for event creators, the purchase of goods or services, most typically associated with a request for quote, proposal, or reverse auction.For bidders, the sale of goods or services.

campus In PeopleSoft Enterprise Campus Solutions, an entity that is usually associated with a distinct physical administrative unit, that belongs to a single academic institution, that uses a unique course catalog, and that produces a common transcript for students within the same academic career.

cash drawer A repository for monies and payments taken locally.

catalog item In PeopleSoft Enterprise Learning Management, a specific topic that a learner can study and have tracked. For example, "Introduction to Microsoft Word." A catalog item contains general information about the topic and includes a course code, description, categorization, keywords, and delivery methods. A catalog item can have one or more learning activities.

catalog map In PeopleSoft Enterprise Catalog Management, translates values from the catalog source data to the format of the company's catalog.

Copyright © 1988-2007, Oracle. All rights reserved.

catalog partner In PeopleSoft Enterprise Catalog Management, shares responsibility with the enterprise catalog manager for maintaining catalog content.

categorization Associates partner offerings with catalog offerings and groups them into enterprise catalog categories.

category In PeopleSoft Enterprise Campus Solutions, a broad grouping to which specific comments or communications (contexts) are assigned. Category codes are also linked to 3C access groups so that you can assign data-entry or view-only privileges across functions.

channel In PeopleSoft MultiChannel Framework, email, chat, voice (computer telephone integration [CTI]), or a generic event.

ChartField A field that stores a chart of accounts, resources, and so on, depending on the PeopleSoft Enterprise application. ChartField values represent individual account numbers, department codes, and so forth.

ChartField balancing You can require specific ChartFields to match up (balance) on the debit and the credit side of a transaction.

ChartField combination edit The process of editing journal lines for valid ChartField combinations based on user-defined rules.

ChartKey One or more fields that uniquely identify each row in a table. Some tables contain only one field as the key, while others require a combination.

checkbook In PeopleSoft Enterprise Promotions Management, enables you to view financial data (such as planned, incurred, and actual amounts) that is related to funds and trade promotions.

checklist code In PeopleSoft Enterprise Campus Solutions, a code that represents a list of planned or completed action items that can be assigned to a staff member, volunteer, or unit. Checklists enable you to view all action assignments on one page.

claimback In the wholesale distribution industry, a contract between supplier and distributor, in which monies are paid to the distributor on the sale of specified products or product groups to targeted customers or customer groups.

class In PeopleSoft Enterprise Campus Solutions, a specific offering of a course component within an academic term.

See also *course.*

Class ChartField A ChartField value that identifies a unique appropriation budget key when you combine it with a fund, department ID, and program code, as well as a budget period. Formerly called *sub-classification.*

clearance In PeopleSoft Enterprise Campus Solutions, the period of time during which a constituent in PeopleSoft Enterprise Contributor Relations is approved for involvement in an initiative or an action. Clearances are used to prevent development officers from making multiple requests to a constituent during the same time period.

clone In PeopleCode, to make a unique copy. In contrast, to *copy* may mean making a new reference to an object, so if the underlying object is changed, both the copy and the original change.

cohort In PeopleSoft Enterprise Campus Solutions, the highest level of the three-level classification structure that you define for enrollment management. You can define a cohort level, link it to other levels, and set enrollment target numbers for it.

See also *population* and *division*.

Copyright © 1988-2007, Oracle. All rights reserved.

collection To make a set of documents available for searching in Verity, you must first create at least one collection. A collection is set of directories and files that allow search application users to use the Verity search engine to quickly find and display source documents that match search criteria. A collection is a set of statistics and pointers to the source documents, stored in a proprietary format on a file server. Because a collection can only store information for a single location, PeopleTools maintains a set of collections (one per language code) for each search index object.

collection rule In PeopleSoft Enterprise Receivables, a user-defined rule that defines actions to take for a customer based on both the amount and the number of days past due for outstanding balances.

comm key See *communication key.*

communication key In PeopleSoft Enterprise Campus Solutions, a single code for entering a combination of communication category, communication context, communication method, communication direction, and standard letter code. Communication keys (also called *comm keys* or *speed keys*) can be created for background processes as well as for specific users.

compensation object In PeopleSoft Enterprise Incentive Management, a node within a compensation structure. Compensation objects are the building blocks that make up a compensation structure's hierarchical representation.

compensation structure In PeopleSoft Enterprise Incentive Management, a hierarchical relationship of compensation objects that represents the compensation-related relationship between the objects.

component interface A component interface is a set of application programming interfaces (APIs) that you can use to access and modify PeopleSoft Enterprise database information using a program instead of the PeopleSoft client.

condition In PeopleSoft Enterprise Receivables, occurs when there is a change of status for a customer's account, such as reaching a credit limit or exceeding a user-defined balance due.

configuration parameter catalog Used to configure an external system with PeopleSoft Enterprise. For example, a configuration parameter catalog might set up configuration and communication parameters for an external server.

configuration plan In PeopleSoft Enterprise Incentive Management, configuration plans hold allocation information for common variables (not incentive rules) and are attached to a node without a participant. Configuration plans are not processed by transactions.

constituents In PeopleSoft Enterprise Campus Solutions, friends, alumni, organizations, foundations, or other entities affiliated with the institution, and about which the institution maintains information. The constituent types delivered with PeopleSoft Enterprise Contributor Relations Solutions are based on those defined by the Council for the Advancement and Support of Education (CASE).

constraint A business policy or rule that affects how a sourcing event is awarded. There are three types of constraints: business, global, and event.

content reference Content references are pointers to content registered in the portal registry. These are typically either URLs or iScripts. Content references fall into three categories: target content, templates, and template pagelets.

context In PeopleCode, determines which buffer fields can be contextually referenced and which is the current row of data on each scroll level when a PeopleCode program is running.

In PeopleSoft Enterprise Campus Solutions, a specific instance of a comment or communication. One or more contexts are assigned to a category, which you link to

 Copyright © 1988-2007, Oracle. All rights reserved.

	3C access groups so that you can assign data-entry or view-only privileges across functions. In PeopleSoft Enterprise Incentive Management, a mechanism that is used to determine the scope of a processing run. PeopleSoft Enterprise Incentive Management uses three types of context: plan, period, and run-level.
control table	Stores information that controls the processing of an application. This type of processing might be consistent throughout an organization, or it might be used only by portions of the organization for more limited sharing of data.
cost plus contract line	A rate-based contract line associated with a fee component of Award, Fixed, Incentive, or Other. Rate-based contract lines associated with a fee type of None are not considered cost-plus contract lines.
cost plus pricing	In PeopleSoft Enterprise Pricer, a pricing method that begins with cost of goods as the basis.
cost profile	A combination of a receipt cost method, a cost flow, and a deplete cost method. A profile is associated with a cost book and determines how items in that book are valued, as well as how the material movement of the item is valued for the book.
cost row	A cost transaction and amount for a set of ChartFields.
counter sale	A face-to-face customer transaction where the customer typically selects items from the storefront or picks up products that they ordered ahead of time. Customers pay for the goods at the counter and take the goods with them instead of having the goods shipped from a warehouse.
course	In PeopleSoft Enterprise Campus Solutions, a course that is offered by a school and that is typically described in a course catalog. A course has a standard syllabus and credit level; however, these may be modified at the class level. Courses can contain multiple components such as lecture, discussion, and lab. See also *class.*
course share set	In PeopleSoft Enterprise Campus Solutions, a tag that defines a set of requirement groups that can share courses. Course share sets are used in PeopleSoft Enterprise Academic Advisement.
current learning	In PeopleSoft Enterprise Learning Management, a self-service repository for all of a learner's in-progress learning activities and programs.
data acquisition	In PeopleSoft Enterprise Incentive Management, the process during which raw business transactions are acquired from external source systems and fed into the operational data store (ODS).
data cube	In PeopleSoft Analytic Calculation Engine, a data cube is a container for one kind of data (such as Sales data) and works with in tandem with one or more dimensions. Dimensions and data cubes in PeopleSoft Analytic Calculation Engine are unrelated to dimensions and online analytical processing (OLAP) cubes in PeopleSoft Cube Manager.
data elements	Data elements, at their simplest level, define a subset of data and the rules by which to group them. For Workforce Analytics, data elements are rules that tell the system what measures to retrieve about your workforce groups.
dataset	A data grouping that enables role-based filtering and distribution of data. You can limit the range and quantity of data that is displayed for a user by associating dataset rules with user roles. The result of dataset rules is a set of data that is appropriate for the user's roles.

Copyright © 1988-2007, Oracle. All rights reserved.

delivery method

In PeopleSoft Enterprise Learning Management, identifies the primary type of delivery method in which a particular learning activity is offered. Also provides default values for the learning activity, such as cost and language. This is primarily used to help learners search the catalog for the type of delivery from which they learn best. Because PeopleSoft Enterprise Learning Management is a blended learning system, it does not enforce the delivery method.

In PeopleSoft Enterprise Supply Chain Management, identifies the method by which goods are shipped to their destinations (such as truck, air, and rail). The delivery method is specified when creating shipment schedules.

delivery method type

In PeopleSoft Enterprise Learning Management, identifies how learning activities can be delivered—for example, through online learning, classroom instruction, seminars, books, and so forth—in an organization. The type determines whether the delivery method includes scheduled components.

detailed business process

A subset of the business process. For example, the detailed business process named Determine Cash Position is a subset of the business process called Cash Management.

dimension

In PeopleSoft Analytic Calculation Engine, a dimension contains a list of one kind of data that can span various contexts, and it is a basic component of an analytic model. Within the analytic model, a dimension is attached to one or more data cubes. In PeopleSoft Cube Manager, a dimension is the most basic component of an OLAP cube and specifies the PeopleSoft metadata to be used to create the dimension's rollup structure. Dimensions and data cubes in PeopleSoft Analytic Calculation Engine are unrelated to dimensions and OLAP cubes in PeopleSoft Cube Manager.

direct receipt

Items shipped from a warehouse or vendor to another warehouse.

direct ship

Items shipped from the vendor or warehouse directly to the customer (formerly referred to as *drop ship*).

directory information tree

In PeopleSoft Enterprise Directory Interface, the representation of a directory's hierarchical structure.

division

In PeopleSoft Enterprise Campus Solutions, the lowest level of the three-level classification structure that you define in PeopleSoft Enterprise Recruiting and Admissions for enrollment management. You can define a division level, link it to other levels, and set enrollment target numbers for it.

See also *population* and *cohort*.

document sequencing

A flexible method that sequentially numbers the financial transactions (for example, bills, purchase orders, invoices, and payments) in the system for statutory reporting and for tracking commercial transaction activity.

dynamic detail tree

A tree that takes its detail values—dynamic details—directly from a table in the database, rather than from a range of values that are entered by the user.

edit table

A table in the database that has its own record definition, such as the Department table. As fields are entered into a PeopleSoft Enterprise application, they can be validated against an edit table to ensure data integrity throughout the system.

effective date

A method of dating information in PeopleSoft Enterprise applications. You can predate information to add historical data to your system, or postdate information in order to enter it before it actually goes into effect. By using effective dates, you don't delete values; you enter a new value with a current effective date.

EIM ledger

Abbreviation for *Enterprise Incentive Management ledger.* In PeopleSoft Enterprise Incentive Management, an object to handle incremental result gathering within the scope of a participant. The ledger captures a result set with all of the appropriate traces to the data origin and to the processing steps of which it is a result.

Copyright © 1988-2007, Oracle. All rights reserved.

elimination set In PeopleSoft Enterprise General Ledger, a related group of intercompany accounts that is processed during consolidations.

entry event In PeopleSoft Enterprise General Ledger, Receivables, Payables, Purchasing, and Billing, a business process that generates multiple debits and credits resulting from single transactions to produce standard, supplemental accounting entries.

equitization In PeopleSoft Enterprise General Ledger, a business process that enables parent companies to calculate the net income of subsidiaries on a monthly basis and adjust that amount to increase the investment amount and equity income amount before performing consolidations.

equity item limit In PeopleSoft Enterprise Campus Solutions, the amounts of funds set by the institution to be awarded with discretionary or gift funds. The limit could be reduced by amounts equal to such things as expected family contribution (EFC) or parent contribution. Students are packaged by Equity Item Type Groups and Related Equity Item Types. This limit can be used to assure that similar student populations are packaged equally.

event A predefined point either in the Component Processor flow or in the program flow. As each point is encountered, the event activates each component, triggering any PeopleCode program that is associated with that component and that event. Examples of events are FieldChange, SavePreChange, and RowDelete.

In PeopleSoft Enterprise Human Resources, also refers to an incident that affects benefits eligibility.

event constraints In PeopleSoft Strategic Sourcing, these constraints are associated with a specific sourcing event. Spend is tracked within the selected event.

event propagation process In PeopleSoft Enterprise Sales Incentive Management, a process that determines, through logic, the propagation of an original PeopleSoft Enterprise Incentive Management event and creates a derivative (duplicate) of the original event to be processed by other objects. PeopleSoft Enterprise Enterprise Sales Incentive Management uses this mechanism to implement splits, roll-ups, and so on. Event propagation determines who receives the credit.

exception In PeopleSoft Enterprise Receivables, an item that either is a deduction or is in dispute.

exclusive pricing In PeopleSoft Enterprise Order Management, a type of arbitration plan that is associated with a price rule. Exclusive pricing is used to price sales order transactions.

fact In PeopleSoft Enterprise applications, facts are numeric data values from fields from a source database as well as an analytic application. A fact can be anything you want to measure your business by, for example, revenue, actual, budget data, or sales numbers. A fact is stored on a fact table.

financial aid term In PeopleSoft Enterprise Campus Solutions, a combination of a period of time that the school determines as an instructional accounting period and an academic career. It is created and defined during the setup process. Only terms eligible for financial aid are set up for each financial aid career.

financial sanctions For U.S. based companies and their foreign subsidiaries, a federal regulation from the Office of Foreign Assets Control (OFAC) requires that vendors be validated against a Specially Designated Nationals (SDN) list prior to payment.

For PeopleSoft Payables, eSettlements, Cash Management, and Order to Cash, you can validate your vendors against any financial sanctions list (for example, the SDN list, a European Union list, and so on).

forecast item A logical entity with a unique set of descriptive demand and forecast data that is used as the basis to forecast demand. You create forecast items for a wide range of uses, but they ultimately represent things that you buy, sell, or use in your organization and for which you require a predictable usage.

Copyright © 1988-2007, Oracle. All rights reserved.

fund In PeopleSoft Enterprise Promotions Management, a budget that can be used to fund promotional activity. There are four funding methods: top down, fixed accrual, rolling accrual, and zero-based accrual.

gap In PeopleSoft Enterprise Campus Solutions, an artificial figure that sets aside an amount of unmet financial aid need that is not funded with Title IV funds. A gap can be used to prevent fully funding any student to conserve funds, or it can be used to preserve unmet financial aid need so that institutional funds can be awarded.

generic process type In PeopleSoft Process Scheduler, process types are identified by a generic process type. For example, the generic process type SQR includes all SQR process types, such as SQR process and SQR report.

gift table In PeopleSoft Enterprise Campus Solutions, a table or so-called *donor pyramid* describing the number and size of gifts that you expect will be needed to successfully complete the campaign in PeopleSoft Enterprise Contributor Relations. The gift table enables you to estimate the number of donors and prospects that you need at each gift level to reach the campaign goal.

GDS Abbreviation for *Global Distribution System.* Broad-based term to describe all computer reservation systems for making travel plans.

GL business unit Abbreviation for *general ledger business unit.* A unit in an organization that is an independent entity for accounting purposes. It maintains its own set of accounting books.

See also *business unit.*

GL entry template Abbreviation for *general ledger entry template.* In PeopleSoft Enterprise Campus Solutions, a template that defines how a particular item is sent to the general ledger. An item-type maps to the general ledger, and the GL entry template can involve multiple general ledger accounts. The entry to the general ledger is further controlled by high-level flags that control the summarization and the type of accounting—that is, accrual or cash.

GL Interface process Abbreviation for *General Ledger Interface process.* In PeopleSoft Enterprise Campus Solutions, a process that is used to send transactions from PeopleSoft Enterprise Student Financials to the general ledger. Item types are mapped to specific general ledger accounts, enabling transactions to move to the general ledger when the GL Interface process is run.

global constraints In PeopleSoft Strategic Sourcing, these constraints apply across multiple Strategic Sourcing business units. Spend is tracked across all of the events from the multiple Strategic Sourcing business units.

group In PeopleSoft Enterprise Billing and Receivables, a posting entity that comprises one or more transactions (items, deposits, payments, transfers, matches, or write-offs).

In PeopleSoft Enterprise Human Resources Management and Supply Chain Management, any set of records that are associated under a single name or variable to run calculations in PeopleSoft business processes. In PeopleSoft Enterprise Time and Labor, for example, employees are placed in groups for time reporting purposes.

ideal response In PeopleSoft Strategic Sourcing, a question that requires the response to match the ideal value for the bid to be considered eligible for award. If the response does not match the ideal value, you can still submit the bid, but it will be disqualified and ineligible for award.

incentive object In PeopleSoft Enterprise Incentive Management, the incentive-related objects that define and support the PeopleSoft Enterprise Incentive Management calculation process and results, such as plan templates, plans, results data, and user interaction objects.

Copyright © 1988-2007, Oracle. All rights reserved.

incentive rule In PeopleSoft Enterprise Sales Incentive Management, the commands that act on transactions and turn them into compensation. A rule is one part in the process of turning a transaction into compensation.

incur In PeopleSoft Enterprise Promotions Management, to become liable for a promotional payment. In other words, you owe that amount to a customer for promotional activities.

initiative In PeopleSoft Enterprise Campus Solutions, the basis from which all advancement plans are executed. It is an organized effort targeting a specific constituency, and it can occur over a specified period of time with specific purposes and goals. An initiative can be a campaign, an event, an organized volunteer effort, a membership drive, or any other type of effort defined by the institution. Initiatives can be multipart, and they can be related to other initiatives. This enables you to track individual parts of an initiative, as well as entire initiatives.

inquiry access In PeopleSoft Enterprise Campus Solutions, a type of security access that permits the user only to view data.

See also *update access.*

institution In PeopleSoft Enterprise Campus Solutions, an entity (such as a university or college) that is independent of other similar entities and that has its own set of rules and business processes.

integration A relationship between two compatible integration points that enables communication to take place between systems. Integrations enable PeopleSoft Enterprise applications to work seamlessly with other PeopleSoft Enterprise applications or with third-party systems or software.

integration point An interface that a system uses to communicate with another PeopleSoft Enterprise application or an external application.

integration set A logical grouping of integrations that applications use for the same business purpose. For example, the integration set ADVANCED_SHIPPING_ORDER contains all of the integrations that notify a customer that an order has shipped.

item In PeopleSoft Enterprise Inventory, a tangible commodity that is stored in a business unit (shipped from a warehouse).

In PeopleSoft Enterprise Demand Planning, Inventory Policy Planning, and Supply Planning, a noninventory item that is designated as being used for planning purposes only. It can represent a family or group of inventory items. It can have a planning bill of material (BOM) or planning routing, and it can exist as a component on a planning BOM. A planning item cannot be specified on a production or engineering BOM or routing, and it cannot be used as a component in a production. The quantity on hand will never be maintained.

In PeopleSoft Enterprise Receivables, an individual receivable. An item can be an invoice, a credit memo, a debit memo, a write-off, or an adjustment.

item shuffle In PeopleSoft Enterprise Campus Solutions, a process that enables you to change a payment allocation without having to reverse the payment.

itinerary In PeopleSoft Expenses, a collection of travel reservations. Itineraries can have reservations that are selected and reserved with the travel vendor. These itineraries are not yet paid for and can be referred to as *pending reservations.* Reservations that have been paid for are referred to as *confirmed reservations.*

joint communication In PeopleSoft Enterprise Campus Solutions, one letter that is addressed jointly to two people. For example, a letter might be addressed to both Mr. Sudhir Awat and Ms. Samantha Mortelli. A relationship must be established between the two individuals in the database, and at least one of the individuals must have an ID in the database.

Copyright © 1988-2007, Oracle. All rights reserved.

keyword In PeopleSoft Enterprise Campus Solutions, a term that you link to particular elements within PeopleSoft Enterprise Student Financials, Financial Aid, and Contributor Relations. You can use keywords as search criteria that enable you to locate specific records in a search dialog box.

KPI An abbreviation for *key performance indicator.* A high-level measurement of how well an organization is doing in achieving critical success factors. This defines the data value or calculation upon which an assessment is determined.

KVI Abbreviation for *Known Value Item.* Term used for products or groups of products where the selling price cannot be reduced or increased.

landlord In PeopleSoft Real Estate Management, an entity that owns real estate and leases the real estate to tenants.

LDIF file Abbreviation for *Lightweight Directory Access Protocol (LDAP) Data Interchange Format file.* Contains discrepancies between PeopleSoft Enterprise data and directory data.

learner group In PeopleSoft Enterprise Learning Management, a group of learners who are linked to the same learning environment. Members of the learner group can share the same attributes, such as the same department or job code. Learner groups are used to control access to and enrollment in learning activities and programs. They are also used to perform group enrollments and mass enrollments in the back office.

learning components In PeopleSoft Enterprise Learning Management, the foundational building blocks of learning activities. PeopleSoft Enterprise Learning Management supports six basic types of learning components: web-based, session, webcast, test, survey, and assignment. One or more of these learning component types compose a single learning activity.

learning environment In PeopleSoft Enterprise Learning Management, identifies a set of categories and catalog items that can be made available to learner groups. Also defines the default values that are assigned to the learning activities and programs that are created within a particular learning environment. Learning environments provide a way to partition the catalog so that learners see only those items that are relevant to them.

learning history In PeopleSoft Enterprise Learning Management, a self-service repository for all of a learner's completed learning activities and programs.

lease In PeopleSoft Real Estate Management, a legally binding agreement between a landlord and a tenant, where the tenant rents all or part of a physical property from the landlord.

lease abstract In PeopleSoft Real Estate Management, a summarized version of the complete lease contract with only the important terms. The lease abstract usually fits on one page and does not include legal terminology.

ledger mapping You use ledger mapping to relate expense data from general ledger accounts to resource objects. Multiple ledger line items can be mapped to one or more resource IDs. You can also use ledger mapping to map dollar amounts (referred to as *rates*) to business units. You can map the amounts in two different ways: an actual amount that represents actual costs of the accounting period, or a budgeted amount that can be used to calculate the capacity rates as well as budgeted model results. In PeopleSoft Enterprise Warehouse, you can map general ledger accounts to the EW Ledger table.

library section In PeopleSoft Enterprise Incentive Management, a section that is defined in a plan (or template) and that is available for other plans to share. Changes to a library section are reflected in all plans that use it.

line In PeopleSoft Strategic Sourcing, an individual item or service upon which there can be a bid.

Copyright © 1988-2007, Oracle. All rights reserved.

linked section In PeopleSoft Enterprise Incentive Management, a section that is defined in a plan template but appears in a plan. Changes to linked sections propagate to plans using that section.

linked variable In PeopleSoft Enterprise Incentive Management, a variable that is defined and maintained in a plan template and that also appears in a plan. Changes to linked variables propagate to plans using that variable.

LMS Abbreviation for *learning management system.* In PeopleSoft Enterprise Campus Solutions, LMS is a PeopleSoft Enterprise Student Records feature that provides a common set of interoperability standards that enable the sharing of instructional content and data between learning and administrative environments.

load In PeopleSoft Enterprise Inventory, identifies a group of goods that are shipped together. Load management is a feature of PeopleSoft Enterprise Inventory that is used to track the weight, the volume, and the destination of a shipment.

local functionality In PeopleSoft Enterprise HRMS, the set of information that is available for a specific country. You can access this information when you click the appropriate country flag in the global window, or when you access it by a local country menu.

location Locations enable you to indicate the different types of addresses—for a company, for example, one address to receive bills, another for shipping, a third for postal deliveries, and a separate street address. Each address has a different location number. The primary location—indicated by a *1*—is the address you use most often and may be different from the main address.

logistical task In PeopleSoft Enterprise Services Procurement, an administrative task that is related to hiring a service provider. Logistical tasks are linked to the service type on the work order so that different types of services can have different logistical tasks. Logistical tasks include both preapproval tasks (such as assigning a new badge or ordering a new laptop) and postapproval tasks (such as scheduling orientation or setting up the service provider email). The logistical tasks can be mandatory or optional. Mandatory preapproval tasks must be completed before the work order is approved. Mandatory postapproval tasks, on the other hand, must be completed before a work order is released to a service provider.

market template In PeopleSoft Enterprise Incentive Management, additional functionality that is specific to a given market or industry and is built on top of a product category.

mass change In PeopleSoft Enterprise Campus Solutions, mass change is a SQL generator that can be used to create specialized functionality. Using mass change, you can set up a series of Insert, Update, or Delete SQL statements to perform business functions that are specific to the institution.

See also *3C engine.*

match group In PeopleSoft Enterprise Receivables, a group of receivables items and matching offset items. The system creates match groups by using user-defined matching criteria for selected field values.

MCF server Abbreviation for *PeopleSoft MultiChannel Framework server.* Comprises the universal queue server and the MCF log server. Both processes are started when *MCF Servers* is selected in an application server domain configuration.

merchandising activity In PeopleSoft Enterprise Promotions Management, a specific discount type that is associated with a trade promotion (such as off-invoice, billback or rebate, or lump-sum payment) that defines the performance that is required to receive the discount. In the industry, you may know this as an offer, a discount, a merchandising event, an event, or a tactic.

Copyright © 1988-2007, Oracle. All rights reserved.

meta-SQL Meta-SQL constructs expand into platform-specific SQL substrings. They are used in functions that pass SQL strings, such as in SQL objects, the SQLExec function, and PeopleSoft Application Engine programs.

metastring Metastrings are special expressions included in SQL string literals. The metastrings, prefixed with a percent (%) symbol, are included directly in the string literals. They expand at run time into an appropriate substring for the current database platform.

multibook In PeopleSoft Enterprise General Ledger, multiple ledgers having multiple-base currencies that are defined for a business unit, with the option to post a single transaction to all base currencies (all ledgers) or to only one of those base currencies (ledgers).

multicurrency The ability to process transactions in a currency other than the business unit's base currency.

national allowance In PeopleSoft Enterprise Promotions Management, a promotion at the corporate level that is funded by nondiscretionary dollars. In the industry, you may know this as a national promotion, a corporate promotion, or a corporate discount.

NDP Abbreviation for *Non-Discountable Products.* Term used for products or groups of products where the selling price cannot be decreased.

need In PeopleSoft Enterprise Campus Solutions, the difference between the cost of attendance (COA) and the expected family contribution (EFC). It is the gap between the cost of attending the school and the student's resources. The financial aid package is based on the amount of financial need. The process of determining a student's need is called *need analysis.*

node-oriented tree A tree that is based on a detail structure, but the detail values are not used.

Optimization Engine A PeopleTools component that Strategic Sourcing leverages to evaluate bids and determine an ideal award allocation. The award recommendation is based on maximizing the value while adhering to purchasing and company objectives and constraints.

pagelet Each block of content on the home page is called a pagelet. These pagelets display summary information within a small rectangular area on the page. The pagelet provide users with a snapshot of their most relevant PeopleSoft Enterprise and non-PeopleSoft Enterprise content.

participant In PeopleSoft Enterprise Incentive Management, participants are recipients of the incentive compensation calculation process.

participant object Each participant object may be related to one or more compensation objects.

See also *compensation object.*

partner A company that supplies products or services that are resold or purchased by the enterprise.

pay cycle In PeopleSoft Enterprise Payables, a set of rules that define the criteria by which it should select scheduled payments for payment creation.

payment shuffle In PeopleSoft Enterprise Campus Solutions, a process allowing payments that have been previously posted to a student's account to be automatically reapplied when a higher priority payment is posted or the payment allocation definition is changed.

pending item In PeopleSoft Enterprise Receivables, an individual receivable (such as an invoice, a credit memo, or a write-off) that has been entered in or created by the system, but hasn't been posted.

Copyright © 1988-2007, Oracle. All rights reserved.

PeopleCode PeopleCode is a proprietary language, executed by the PeopleSoft Enterprise component processor. PeopleCode generates results based on existing data or user actions. By using various tools provided with PeopleTools, external services are available to all PeopleSoft Enterprise applications wherever PeopleCode can be executed.

PeopleCode event See *event.*

PeopleSoft Pure Internet Architecture The fundamental architecture on which PeopleSoft 8 applications are constructed, consisting of a relational database management system (RDBMS), an application server, a web server, and a browser.

performance measurement In PeopleSoft Enterprise Incentive Management, a variable used to store data (similar to an aggregator, but without a predefined formula) within the scope of an incentive plan. Performance measures are associated with a plan calendar, territory, and participant. Performance measurements are used for quota calculation and reporting.

period context In PeopleSoft Enterprise Incentive Management, because a participant typically uses the same compensation plan for multiple periods, the period context associates a plan context with a specific calendar period and fiscal year. The period context references the associated plan context, thus forming a chain. Each plan context has a corresponding set of period contexts.

person of interest A person about whom the organization maintains information but who is not part of the workforce.

personal portfolio In PeopleSoft Enterprise Campus Solutions, the user-accessible menu item that contains an individual's name, address, telephone number, and other personal information.

phase A level 1 task, meaning that if a task had subtasks, the level 1 task would be considered the phase.

pickup quantity The product quantity that the customer is taking with them from the counter sales environment.

plan In PeopleSoft Enterprise Sales Incentive Management, a collection of allocation rules, variables, steps, sections, and incentive rules that instruct the PeopleSoft Enterprise Incentive Management engine in how to process transactions.

plan context In PeopleSoft Enterprise Incentive Management, correlates a participant with the compensation plan and node to which the participant is assigned, enabling the PeopleSoft Enterprise Incentive Management system to find anything that is associated with the node and that is required to perform compensation processing. Each participant, node, and plan combination represents a unique plan context—if three participants are on a compensation structure, each has a different plan context. Configuration plans are identified by plan contexts and are associated with the participants that refer to them.

plan template In PeopleSoft Enterprise Incentive Management, the base from which a plan is created. A plan template contains common sections and variables that are inherited by all plans that are created from the template. A template may contain steps and sections that are not visible in the plan definition.

planned learning In PeopleSoft Enterprise Learning Management, a self-service repository for all of a learner's planned learning activities and programs.

planning instance In PeopleSoft Enterprise Supply Planning, a set of data (business units, items, supplies, and demands) constituting the inputs and outputs of a supply plan.

population In PeopleSoft Enterprise Campus Solutions, the middle level of the three-level classification structure that you define in PeopleSoft Enterprise Recruiting and

Copyright © 1988-2007, Oracle. All rights reserved.

	Admissions for enrollment management. You can define a population level, link it to other levels, and set enrollment target numbers for it. See also *division* and *cohort.*
portal registry	In PeopleSoft Enterprise applications, the portal registry is a tree-like structure in which content references are organized, classified, and registered. It is a central repository that defines both the structure and content of a portal through a hierarchical, tree-like structure of folders useful for organizing and securing content references.
predecessor task	A task that you must complete before you start another task.
price breaks	In PeopleSoft Strategic Sourcing, a price discount or surcharge that a bidder may apply based on the quantity awarded.
price components	In PeopleSoft Strategic Sourcing, the various components, such as material costs, labor costs, shipping costs, and so on that make up the overall bid price.
price list	Enables you to select products and conditions for which the price list applies to a transaction. During a transaction, the system either determines the product price based on the predefined search hierarchy for the transaction or uses the product's lowest price on any associated, active price lists. This price is used as the basis for any further discounts and surcharges.
price rule	The conditions that must be met for adjustments to be applied to the base price. Multiple rules can apply when conditions of each rule are met.
price rule conditions	Conditions that select the price-by fields, the values for the price-by fields, and the operator that determines how the price-by fields relate to the transaction.
price rule key	The fields that are available to define price rule conditions (which are used to match a transaction) on the price rule.
primacy number	In PeopleSoft Enterprise Campus Solutions, a number that the system uses to prioritize financial aid applications when students are enrolled in multiple academic careers and academic programs at the same time. The Consolidate Academic Statistics process uses the primacy number indicated for both the career and program at the institutional level to determine a student's primary career and program. The system also uses the number to determine the primary student attribute value that is used when you extract data to report on cohorts. The lowest number takes precedence.
primary name type	In PeopleSoft Enterprise Campus Solutions, the name type that is used to link the name stored at the highest level within the system to the lower-level set of names that an individual provides.
process category	In PeopleSoft Process Scheduler, processes that are grouped for server load balancing and prioritization.
process group	In PeopleSoft Enterprise Financials, a group of application processes (performed in a defined order) that users can initiate in real time, directly from a transaction entry page.
process definition	Process definitions define each run request.
process instance	A unique number that identifies each process request. This value is automatically incremented and assigned to each requested process when the process is submitted to run.
process job	You can link process definitions into a job request and process each request serially or in parallel. You can also initiate subsequent processes based on the return code from each prior request.

Copyright © 1988-2007, Oracle. All rights reserved.

process request A single run request, such as a Structured Query Report (SQR), a COBOL or Application Engine program, or a Crystal report that you run through PeopleSoft Process Scheduler.

process run control A PeopleTools variable used to retain PeopleSoft Process Scheduler values needed at runtime for all requests that reference a run control ID. Do not confuse these with application run controls, which may be defined with the same run control ID, but only contain information specific to a given application process request.

product A PeopleSoft Enterprise or third-party product. PeopleSoft organizes its software products into product families and product lines. Interactive Services Repository contains information about every release of every product that PeopleSoft sells, as well as products from certified third-party companies. These products appear with the product name and release number.

product adds The pricing functionality where buying product A gets product B for free or at a price (formerly referred to as *giveaways*).

product bidding In PeopleSoft Strategic Sourcing, the placing of a bid on behalf of the bidder, up or down to the bidder's specified amount, so that the bidder can be the leading bidder.

product category In PeopleSoft Enterprise Incentive Management, indicates an application in the PeopleSoft Enterprise Incentive Management suite of products. Each transaction in the PeopleSoft Enterprise Incentive Management system is associated with a product category.

product family A group of products that are related by common functionality. The family names that can be searched using Interactive Service Repository are Oracle's PeopleSoft Enterprise, PeopleSoft EnterpriseOne, PeopleSoft World, and third-party, certified partners.

product line The name of a PeopleSoft Enterprise product line or the company name of a third-party certified partner. Integration Services Repository enables you to search for integration points by product line.

programs In PeopleSoft Enterprise Learning Management, a high-level grouping that guides the learner along a specific learning path through sections of catalog items. PeopleSoft Enterprise Learning Systems provides two types of programs—curricula and certifications.

progress log In PeopleSoft Enterprise Services Procurement, tracks deliverable-based projects. This is similar to the time sheet in function and process. The service provider contact uses the progress log to record and submit progress on deliverables. The progress can be logged by the activity that is performed, by the percentage of work that is completed, or by the completion of milestone activities that are defined for the project.

project transaction In PeopleSoft Enterprise Project Costing, an individual transaction line that represents a cost, time, budget, or other transaction row.

promotion In PeopleSoft Enterprise Promotions Management, a trade promotion, which is typically funded from trade dollars and used by consumer products manufacturers to increase sales volume.

prospects In PeopleSoft Enterprise Campus Solutions, students who are interested in applying to the institution.

In PeopleSoft Enterprise Contributor Relations, individuals and organizations that are most likely to make substantial financial commitments or other types of commitments to the institution.

proxy bidding In PeopleSoft Strategic Sourcing, the placing of a bid on behalf of the bidder, up or down to the bidder's specified amount, so that the bidder can be the leading bidder.

Copyright © 1988-2007, Oracle. All rights reserved.

publishing In PeopleSoft Enterprise Incentive Management, a stage in processing that makes incentive-related results available to participants.

rating components In PeopleSoft Enterprise Campus Solutions, variables used with the Equation Editor to retrieve specified populations.

record group A set of logically and functionally related control tables and views. Record groups help enable TableSet sharing, which eliminates redundant data entry. Record groups ensure that TableSet sharing is applied consistently across all related tables and views.

record input VAT flag Abbreviation for *record input value-added tax flag.* Within PeopleSoft Enterprise Purchasing, Payables, and General Ledger, this flag indicates that you are recording input VAT on the transaction. This flag, in conjunction with the record output VAT flag, is used to determine the accounting entries created for a transaction and to determine how a transaction is reported on the VAT return. For all cases within Purchasing and Payables where VAT information is tracked on a transaction, this flag is set to Yes. This flag is not used in PeopleSoft Enterprise Order Management, Billing, or Receivables, where it is assumed that you are always recording only output VAT, or in PeopleSoft Enterprise Expenses, where it is assumed that you are always recording only input VAT.

record output VAT flag Abbreviation for *record output value-added tax flag.*

See *record input VAT flag.*

recname The name of a record that is used to determine the associated field to match a value or set of values.

recognition In PeopleSoft Enterprise Campus Solutions, the recognition type indicates whether the PeopleSoft Enterprise Contributor Relations donor is the primary donor of a commitment or shares the credit for a donation. Primary donors receive hard credit that must total 100 percent. Donors that share the credit are given soft credit. Institutions can also define other share recognition-type values such as memo credit or vehicle credit.

reference data In PeopleSoft Enterprise Sales Incentive Management, system objects that represent the sales organization, such as territories, participants, products, customers, and channels.

reference object In PeopleSoft Enterprise Incentive Management, this dimension-type object further defines the business. Reference objects can have their own hierarchy (for example, product tree, customer tree, industry tree, and geography tree).

reference transaction In commitment control, a reference transaction is a source transaction that is referenced by a higher-level (and usually later) source transaction, in order to automatically reverse all or part of the referenced transaction's budget-checked amount. This avoids duplicate postings during the sequential entry of the transaction at different commitment levels. For example, the amount of an encumbrance transaction (such as a purchase order) will, when checked and recorded against a budget, cause the system to concurrently reference and relieve all or part of the amount of a corresponding pre-encumbrance transaction, such as a purchase requisition.

regional sourcing In PeopleSoft Enterprise Purchasing, provides the infrastructure to maintain, display, and select an appropriate vendor and vendor pricing structure that is based on a regional sourcing model where the multiple ship to locations are grouped. Sourcing may occur at a level higher than the ship to location.

relationship object In PeopleSoft Enterprise Incentive Management, these objects further define a compensation structure to resolve transactions by establishing associations between compensation objects and business objects.

remote data source data Data that is extracted from a separate database and migrated into the local database.

Copyright © 1988-2007, Oracle. All rights reserved.

REN server Abbreviation for *real-time event notification server* in PeopleSoft MultiChannel Framework.

requester In PeopleSoft Enterprise eSettlements, an individual who requests goods or services and whose ID appears on the various procurement pages that reference purchase orders.

reservations In PeopleSoft Expenses, travel reservations that have been placed with the travel vendor.

reversal indicator In PeopleSoft Enterprise Campus Solutions, an indicator that denotes when a particular payment has been reversed, usually because of insufficient funds.

RFI event In PeopleSoft Strategic Sourcing, a request for information.

RFx event In PeopleSoft Strategic Sourcing, a request for proposal or request for a quote event when bidders submit their overall best bids and during which bidders do not actively compete against one another.

role Describes how people fit into PeopleSoft Workflow. A role is a class of users who perform the same type of work, such as clerks or managers. Your business rules typically specify what user role needs to do an activity.

role user A PeopleSoft Workflow user. A person's role user ID serves much the same purpose as a user ID does in other parts of the system. PeopleSoft Workflow uses role user IDs to determine how to route worklist items to users (through an email address, for example) and to track the roles that users play in the workflow. Role users do not need PeopleSoft user IDs.

roll up In a tree, to roll up is to total sums based on the information hierarchy.

run control A run control is a type of online page that is used to begin a process, such as the batch processing of a payroll run. Run control pages generally start a program that manipulates data.

run control ID A unique ID to associate each user with his or her own run control table entries.

run-level context In PeopleSoft Enterprise Incentive Management, associates a particular run (and batch ID) with a period context and plan context. Every plan context that participates in a run has a separate run-level context. Because a run cannot span periods, only one run-level context is associated with each plan context.

saved bid In PeopleSoft Strategic Sourcing, a bid that has been created but not submitted. Only submitted bids are eligible for award.

score In PeopleSoft Strategic Sourcing, the numerical sum of answers (percentages) to bid factors on an event. Scores appear only to bidders on auction events.

SCP SCBM XML message Abbreviation for *Supply Chain Planning Supply Chain Business Modeler Extensible Markup Language message.* Supply Chain Business Modeler uses XML as the format for all data that it imports and exports.

search query You use this set of objects to pass a query string and operators to the search engine. The search index returns a set of matching results with keys to the source documents.

search/match In PeopleSoft Enterprise Campus Solutions and PeopleSoft Enterprise Human Resources Management Solutions, a feature that enables you to search for and identify duplicate records in the database.

seasonal address In PeopleSoft Enterprise Campus Solutions, an address that recurs for the same length of time at the same time of year each year until adjusted or deleted.

Copyright © 1988-2007, Oracle. All rights reserved.

section In PeopleSoft Enterprise Incentive Management, a collection of incentive rules that operate on transactions of a specific type. Sections enable plans to be segmented to process logical events in different sections.

security event In commitment control, security events trigger security authorization checking, such as budget entries, transfers, and adjustments; exception overrides and notifications; and inquiries.

sell event In PeopleSoft Strategic Sourcing, for event creators, the sale of goods or services most typically associated with forward auctions. For bidders, the purchase of goods or services.

serial genealogy In PeopleSoft Enterprise Manufacturing, the ability to track the composition of a specific, serial-controlled item.

serial in production In PeopleSoft Enterprise Manufacturing, enables the tracing of serial information for manufactured items. This is maintained in the Item Master record.

service impact In PeopleSoft Enterprise Campus Solutions, the resulting action triggered by a service indicator. For example, a service indicator that reflects nonpayment of account balances by a student might result in a service impact that prohibits registration for classes.

service indicator In PeopleSoft Enterprise Campus Solutions, indicates services that may be either withheld or provided to an individual. Negative service indicators indicate holds that prevent the individual from receiving specified services, such as check-cashing privileges or registration for classes. Positive service indicators designate special services that are provided to the individual, such as front-of-line service or special services for disabled students.

session In PeopleSoft Enterprise Campus Solutions, time elements that subdivide a term into multiple time periods during which classes are offered. In PeopleSoft Enterprise Contributor Relations, a session is the means of validating gift, pledge, membership, or adjustment data entry . It controls access to the data entered by a specific user ID. Sessions are balanced, queued, and then posted to the institution's financial system. Sessions must be posted to enter a matching gift or pledge payment, to make an adjustment, or to process giving clubs or acknowledgements.

In PeopleSoft Enterprise Learning Management, a single meeting day of an activity (that is, the period of time between start and finish times within a day). The session stores the specific date, location, meeting time, and instructor. Sessions are used for scheduled training.

session template In PeopleSoft Enterprise Learning Management, enables you to set up common activity characteristics that may be reused while scheduling a PeopleSoft Enterprise Learning Management activity—characteristics such as days of the week, start and end times, facility and room assignments, instructors, and equipment. A session pattern template can be attached to an activity that is being scheduled. Attaching a template to an activity causes all of the default template information to populate the activity session pattern.

setup relationship In PeopleSoft Enterprise Incentive Management, a relationship object type that associates a configuration plan with any structure node.

share driver expression In PeopleSoft Enterprise Business Planning, a named planning method similar to a driver expression, but which you can set up globally for shared use within a single planning application or to be shared between multiple planning applications through PeopleSoft Enterprise Warehouse.

short-term customer A customer not in the system who is entered during sales order entry using a template.

Copyright © 1988-2007, Oracle. All rights reserved.

single signon With single signon, users can, after being authenticated by a PeopleSoft Enterprise application server, access a second PeopleSoft Enterprise application server without entering a user ID or password.

source key process In PeopleSoft Enterprise Campus Solutions, a process that relates a particular transaction to the source of the charge or financial aid. On selected pages, you can drill down into particular charges.

source transaction In commitment control, any transaction generated in a PeopleSoft Enterprise or third-party application that is integrated with commitment control and which can be checked against commitment control budgets. For example, a pre-encumbrance, encumbrance, expenditure, recognized revenue, or collected revenue transaction.

sourcing objective For constraints, the option to designate whether a business rule is required (mandatory) or is only recommended (target).

speed key See *communication key.*

SpeedChart A user-defined shorthand key that designates several ChartKeys to be used for voucher entry. Percentages can optionally be related to each ChartKey in a SpeedChart definition.

SpeedType A code representing a combination of ChartField values. SpeedTypes simplify the entry of ChartFields commonly used together.

staging A method of consolidating selected partner offerings with the offerings from the enterprise's other partners.

standard letter code In PeopleSoft Enterprise Campus Solutions, a standard letter code used to identify each letter template available for use in mail merge functions. Every letter generated in the system must have a standard letter code identification.

statutory account Account required by a regulatory authority for recording and reporting financial results. In PeopleSoft Enterprise, this is equivalent to the Alternate Account (ALTACCT) ChartField.

step In PeopleSoft Enterprise Sales Incentive Management, a collection of sections in a plan. Each step corresponds to a step in the job run.

storage level In PeopleSoft Enterprise Inventory, identifies the level of a material storage location. Material storage locations are made up of a business unit, a storage area, and a storage level. You can set up to four storage levels.

subcustomer qualifier A value that groups customers into a division for which you can generate detailed history, aging, events, and profiles.

Summary ChartField You use summary ChartFields to create summary ledgers that roll up detail amounts based on specific detail values or on selected tree nodes. When detail values are summarized using tree nodes, summary ChartFields must be used in the summary ledger data record to accommodate the maximum length of a node name (20 characters).

summary ledger An accounting feature used primarily in allocations, inquiries, and PS/nVision reporting to store combined account balances from detail ledgers. Summary ledgers increase speed and efficiency of reporting by eliminating the need to summarize detail ledger balances each time a report is requested. Instead, detail balances are summarized in a background process according to user-specified criteria and stored on summary ledgers. The summary ledgers are then accessed directly for reporting.

summary time period In PeopleSoft Enterprise Business Planning, any time period (other than a base time period) that is an aggregate of other time periods, including other summary time periods and base time periods, such as quarter and year total.

Copyright © 1988-2007, Oracle. All rights reserved.

summary tree A tree used to roll up accounts for each type of report in summary ledgers. Summary trees enable you to define trees on trees. In a summary tree, the detail values are really nodes on a detail tree or another summary tree (known as the *basis* tree). A summary tree structure specifies the details on which the summary trees are to be built.

syndicate To distribute a production version of the enterprise catalog to partners.

system function In PeopleSoft Enterprise Receivables, an activity that defines how the system generates accounting entries for the general ledger.

system source The system source identifies the source of a transaction row in the database. For example, a transaction that originates in PeopleSoft Enterprise Expenses contains a system source code of BEX (Expenses Batch).

When PeopleSoft Enterprise Project Costing prices the source transaction row for billing, the system creates a new row with a system source code of PRP (Project Costing pricing), which represents the system source of the new row. System source codes can identify sources that are internal or external to the PeopleSoft Enterprise system. For example, processes that import data from Microsoft Project into PeopleSoft Enterprise applications create transaction rows with a source code of MSP (Microsoft Project).

TableSet A means of sharing similar sets of values in control tables, where the actual data values are different but the structure of the tables is the same.

TableSet sharing Shared data that is stored in many tables that are based on the same TableSets. Tables that use TableSet sharing contain the SETID field as an additional key or unique identifier.

target currency The value of the entry currency or currencies converted to a single currency for budget viewing and inquiry purposes.

task A deliverable item on the detailed sourcing plan.

tax authority In PeopleSoft Enterprise Campus Solutions, a user-defined element that combines a description and percentage of a tax with an account type, an item type, and a service impact.

template A template is HTML code associated with a web page. It defines the layout of the page and also where to get HTML for each part of the page. In PeopleSoft Enterprise, you use templates to build a page by combining HTML from a number of sources. For a PeopleSoft Enterprise portal, all templates must be registered in the portal registry, and each content reference must be assigned a template.

tenant In PeopleSoft Real Estate Management, an entity that leases real estate from a landlord.

territory In PeopleSoft Enterprise Sales Incentive Management, hierarchical relationships of business objects, including regions, products, customers, industries, and participants.

third party A company or vendor that has extensive PeopleSoft Enterprise product knowledge and whose products and integrations have been certified and are compatible with PeopleSoft Enterprise applications.

tiered pricing Enables different portions of a schedule to be priced differently from one another.

time span A relative period, such as year-to-date or current period, that various PeopleSoft General Ledger functions and reports can use when a rolling time frame, rather than a specific date, is required.

total cost In PeopleSoft Strategic Sourcing, the estimated dollar cost (sum of real price dollars and potential “soft” or non-price dollars) of a particular award approach.

Copyright © 1988-2007, Oracle. All rights reserved.

travel group In PeopleSoft Expenses, the organization's travel rules and polices that are associated with specific business units, departments, or employees. You must define at least one travel group when setting up the PeopleSoft Expenses travel feature. You must define and associate at least one travel group with a travel vendor.

travel partner In PeopleSoft Expenses, the travel vendor with which the organization has a contractual relationship.

3C engine Abbreviation for *Communications, Checklists, and Comments engine.* In PeopleSoft Enterprise Campus Solutions, the 3C engine enables you to automate business processes that involve additions, deletions, and updates to communications, checklists, and comments. You define events and triggers to engage the engine, which runs the mass change and processes the 3C records (for individuals or organizations) immediately and automatically from within business processes.

3C group Abbreviation for *Communications, Checklists, and Comments group.* In PeopleSoft Enterprise Campus Solutions, a method of assigning or restricting access privileges. A 3C group enables you to group specific communication categories, checklist codes, and comment categories. You can then assign the group inquiry-only access or update access, as appropriate.

trace usage In PeopleSoft Enterprise Manufacturing, enables the control of which components will be traced during the manufacturing process. Serial- and lot-controlled components can be traced. This is maintained in the Item Master record.

transaction allocation In PeopleSoft Enterprise Incentive Management, the process of identifying the owner of a transaction. When a raw transaction from a batch is allocated to a plan context, the transaction is duplicated in the PeopleSoft Enterprise Incentive Management transaction tables.

transaction state In PeopleSoft Enterprise Incentive Management, a value assigned by an incentive rule to a transaction. Transaction states enable sections to process only transactions that are at a specific stage in system processing. After being successfully processed, transactions may be promoted to the next transaction state and "picked up" by a different section for further processing.

Translate table A system edit table that stores codes and translate values for the miscellaneous fields in the database that do not warrant individual edit tables of their own.

tree The graphical hierarchy in PeopleSoft Enterprise systems that displays the relationship between all accounting units (for example, corporate divisions, projects, reporting groups, account numbers) and determines roll-up hierarchies.

tuition lock In PeopleSoft Enterprise Campus Solutions, a feature in the Tuition Calculation process that enables you to specify a point in a term after which students are charged a minimum (or *locked*) fee amount. Students are charged the locked fee amount even if they later drop classes and take less than the normal load level for that tuition charge.

unclaimed transaction In PeopleSoft Enterprise Incentive Management, a transaction that is not claimed by a node or participant after the allocation process has completed, usually due to missing or incomplete data. Unclaimed transactions may be manually assigned to the appropriate node or participant by a compensation administrator.

universal navigation header Every PeopleSoft Enterprise portal includes the universal navigation header, intended to appear at the top of every page as long as the user is signed on to the portal. In addition to providing access to the standard navigation buttons (like Home, Favorites, and signoff) the universal navigation header can also display a welcome message for each user.

update access In PeopleSoft Enterprise Campus Solutions, a type of security access that permits the user to edit and update data.

Copyright © 1988-2007, Oracle. All rights reserved.

	See also *inquiry access*.
user interaction object	In PeopleSoft Enterprise Sales Incentive Management, used to define the reporting components and reports that a participant can access in his or her context. All PeopleSoft Enterprise Sales Incentive Management user interface objects and reports are registered as user interaction objects. User interaction objects can be linked to a compensation structure node through a compensation relationship object (individually or as groups).
variable	In PeopleSoft Enterprise Sales Incentive Management, the intermediate results of calculations. Variables hold the calculation results and are then inputs to other calculations. Variables can be plan variables that persist beyond the run of an engine or local variables that exist only during the processing of a section.
VAT exception	Abbreviation for *value-added tax exception.* A temporary or permanent exemption from paying VAT that is granted to an organization. This terms refers to both VAT exoneration and VAT suspension.
VAT exempt	Abbreviation for *value-added tax exempt.* Describes goods and services that are not subject to VAT. Organizations that supply exempt goods or services are unable to recover the related input VAT. This is also referred to as exempt without recovery.
VAT exoneration	Abbreviation for *value-added tax exoneration.* An organization that has been granted a permanent exemption from paying VAT due to the nature of that organization.
VAT suspension	Abbreviation for *value-added tax suspension.* An organization that has been granted a temporary exemption from paying VAT.
warehouse	A PeopleSoft Enterprise data warehouse that consists of predefined ETL maps, data warehouse tools, and DataMart definitions.
weight or weighting	In PeopleSoft Strategic Sourcing, how important the line or question is to the overall event. Weighting is used to score and analyze bids. For RFx and RFI events, weightings may or may not appear to bidders.
work order	In PeopleSoft Enterprise Services Procurement, enables an enterprise to create resource-based and deliverable-based transactions that specify the basic terms and conditions for hiring a specific service provider. When a service provider is hired, the service provider logs time or progress against the work order.
worker	A person who is part of the workforce; an employee or a contingent worker.
workset	A group of people and organizations that are linked together as a set. You can use worksets to simultaneously retrieve the data for a group of people and organizations and work with the information on a single page.
worksheet	A way of presenting data through a PeopleSoft Enterprise Business Analysis Modeler interface that enables users to do in-depth analysis using pivoting tables, charts, notes, and history information.
worklist	The automated to-do list that PeopleSoft Workflow creates. From the worklist, you can directly access the pages you need to perform the next action, and then return to the worklist for another item.
XML link	The XML Linking language enables you to insert elements into XML documents to create a links between resources.
XML schema	An XML definition that standardizes the representation of application messages, component interfaces, or business interlinks.
XPI	Abbreviation for *eXtended Process Integrator.* PeopleSoft XPI is the integration infrastructure that enables both real-time and batch communication with JD Edwards EnterpriseOne applications.

Copyright © 1988-2007, Oracle. All rights reserved.

yield by operation In PeopleSoft Enterprise Manufacturing, the ability to plan the loss of a manufactured item on an operation-by-operation basis.

zero-rated VAT Abbreviation for *zero-rated value-added tax.* A VAT transaction with a VAT code that has a tax percent of zero. Used to track taxable VAT activity where no actual VAT amount is charged. Organizations that supply zero-rated goods and services can still recover the related input VAT. This is also referred to as exempt with recovery.

Copyright © 1988-2007, Oracle. All rights reserved.

Copyright © 1988-2007, Oracle. All rights reserved.

Index

A

actions, inserting into transform programs 379
adapters, iWay SOAPswitch, *See* ERP connectors
Add a New Record page 182
additional documentation xxxvi
aliases
  fields 182
  issuer aliases for root certificates 594
  preserving for records and fields 383
  records 182
  routing definition 334
  specifying field name 185
  specifying record aliases (rowset-based messages) 184
  understanding 96
any-to-local routing
  defined 331
  generating 336
  regenerating 337
any-to-local routing definitions
  viewing status 336
Apache Xalan 378, 773
Apache Xerces 773
application classes
  error-handling methods 217
Application Designer
  defining transform programs 375
Application Engine transform programs, *See* transformations
application fundamentals xxxv
application servers
  applying transformations 74
  generating requests 17
  handling exceptions 504
  logging 503
  Publication Broker 19
    See Also Publication Broker
  Publication Contractor 19
    See Also Publication Contractor
  receiving requests 16
  returning responses 16
  Subscription Contractor 19
    See Also Subscription Contractor
  tracing 503
archiving
  asynchronous service operations 432
  deleting archived service operations 493
  running batch service operation archiving processes 455
  searching for archived service operations 420, 436
  service operation instances 453
  service operations assigned to a queue 198
  setting error logging properties 73
  setting system message logging properties 73
AS2
  target connectors 119
    See Also AS2 target connectors
AS2 connectors
  listening connectors 117
    See Also AS2 listening connectors
AS2 listening connectors
  understanding 117
AS2 target connectors
  understanding 119
AsyncFFsend 554
asynchronous
  integrations 41
    See Also asynchronous integrations
  messaging/transmission 41
    See Also asynchronous messaging
Asynchronous – One Way 292
asynchronous integrations
  configuring messaging servers 41
asynchronous message transmission
  handling outbound 223
asynchronous messaging
  brokers, contractors, queues 19
  debugging transformations 504
  handling cookies 227
  monitoring service operation details 427
    See Also Message Details component
  partitioning channels 199
    See Also partitioning

Copyright © 1988-2007, Oracle. All rights reserved.

publishing 22
refreshing queues 50
responding to requests 134
running batch error notifications 450
server processes 20
simulating receiving messages from external nodes 250
submitting non-SOAP messages 136
understanding compression 94
understanding dedicated servers 43
understanding subscription 25
using JMS target connectors 150
viewing errors 443
viewing in XML format 446
viewing inbound post statistics 465
viewing nonrepudiation information 449
viewing performance statistics 457
Asynchronous Request/Response 292
Asynchronous Services component
monitoring asynchronous service operation instances 423
Asynchronous to Synchronous 292
asynchronous transactions
handling inbound 232
iWay SOAPswitch operations 573
See Also iWay SOAPswitch
audit action codes 100
authentication
authenticating nodes 364
digital certificates 588
See Also digital certificates
setting for FTPS connectors 160
authorization
running HTTP transactions through proxy servers 137

B

Baltimore Technologies 588
batch operations
running error notifications 450
running service operation archiving processes 455
BEA Jolt, *See* Jolt
BEA Tuxedo, *See* Tuxedo
BEA WebLogic, *See* WebLogic
BPEL integrations
BPEL node 555
consuming asynchronous fire-and-forget BPEL operations 562
consuming asynchronous one-way BPEL operations 562
consuming asynchronous request/response BPEL operations 559
consuming BPEL process-based services 556
consuming synchronous BPEL operations 557
delivered application classes for 553
monitoring 554
prerequisites 554
providing asynchronous PeopleSoft request/response to BPEL processes 568
providing PeopleSoft services to BPEL processes 564
providing synchronous PeopleSoft operations to BPEL processes 565
BPEL processes 553
See Also BPEL integrations
understanding integrating with 553
BPELUtil 554
broker dispatchers/handlers
asynchronous publishing of instances 22
asynchronous subscription of instances 25
server processes 21
understanding 458
viewing statistics 466
Broker Handler page 466
built-in functions
bypassing the integration engine to send messages 78
ConnectorRequest 78
ConnectorRequestURL 78
Exit 252
FindCodeSetValues 410
PublishXmlDoc 689
SetXMLDoc 106

C

C/C++
building DLLs for native library 774
creating JNI headers/header functions 774
creating target connectors 774
developing Java target connectors 773
registering DLLs for native library 774

Copyright © 1988-2007, Oracle. All rights reserved.

understanding connector development 773
CA
accessing certificate properties 596
adding CA authorities 592
installing application server-based digital certificates 592
installing root certificates 592
obtaining signed public encryption keys 590
resolving root certificate mismatches 595
understanding 588
understanding root certificates 590
CDATA
generating outbound messages 222
handling non-XML data 381
passing HTTP parameters 130
sending non-XML files 106
submitting non-XML messages 130
understanding 129
certificate authorities (CA), *See* CA
certificate signing request (CSR), *See* CSR
certificates
certificate authorities (CA) 588
See Also CA
certificate signing request (CSR) 590
See Also CSR
digital 588
See Also digital certificates
public key 590
See Also public key certificates
root 590
See Also root certificates
setting security properties 67
setting up local node 593
setting up remote node 595
channels
debugging 504
handling nonrepudiation messages 134
setting subchannels for JMS queue listeners 144
setting subchannels for JMS topic subscribers 146
classes
developing for connectors 756
developing for listening connectors 761
developing for target connectors 757
installing connector classes 765
Message 96, 99, 105, 251
Rowset 96, 105, 129, 383
SoapDoc 105, 106, 177, 178, 249, 251
TransformData 380, 381
using Java XML DOM wrapper classes 770
XmlDoc 99, 105, 177, 251, 395
clean-up mode 479
client authentication
generating private and public key pairs 599
Codeset Group page 400
codeset groups
defining 400
selecting for nodes 366
understanding 399
Codeset page 402
codeset repositories
elements 399
translating via 399
codeset values
defining 402
deleting 404
understanding 400
Codeset Values page 402
codesets
codeset groups 399
See Also codeset groups
codeset values 400
See Also codeset values
defining 402
deleting 404
exporting/importing 404
repository 399
See Also codeset repositories
understanding 400
comments, submitting xl
common elements xl
component interface-based services 321
generating 327
impact of changing component interfaces 323
metadata created 321
naming conventions of metadata created 321
prerequisites for creating 323
selecting component interfaces methods to include as service operations 324
selecting component interfaces to expose as services 323
user-defined method restrictions 322

Copyright © 1988-2007, Oracle. All rights reserved.

viewing generated service definitions 328
components
Gateways component 54
See Also Gateways component
iWay SOAPswitch 572
compression
applying transformations at gateway 74
setting for FTP target connectors 158
setting for HTTP connectors 128
setting for JMS messages 152
setting for PeopleSoft 8.1 listening connectors 140
setting for simple file target connectors 156
setting for SMTP target connectors 175
setting message properties 121
setting the message compression threshold 94
Connector Properties page 58
connector SDK
API documentation 752
contents 752
developing connectors 753
developing connectors based on DOM 770
developing connectors in C/C++ 773
laoding connectors 57
locating SDK and its contents 752
reusing connector code 777
understanding 8, 751
connectors
configuring integration gateways 54
connector management service 10
developing 753
developing based on DOM 770
developing classes for 756
developing in C/C++ environment 773
ERP 572
See Also ERP connectors
exchanging input/output formats 756
incoming request flow through the architecture 14
installing classes 765
interacting between local and external systems 757
listening 8
See Also listening connectors
outgoing request flow through the architecture 14
receiving third-party messages 700
reusing code 777
SDK 751
See Also connector SDK
setting gateway properties 69
target 8
See Also target connectors
understanding 8
understanding delivered 115
understanding gateway manager 9
See Also gateway manager
using templates 766
Consume Web Service wizard
accessing metadata created 549
asynchronous request and response operations 536
binding style of consumed WSDL documents 535
converting asynchronous operations 542
features 533
metadata created 534
multiple fault messages 535
multiple root elements 535
operation types supported 533
prerequisites 536
renaming operation messages 544
selecting queues for asynchronous operations 546
selecting receiver nodes 547
selecting service operations 542
selecting service ports 541
selecting services 541
selecting WSDL sources 539
sources for consuming WSDL documents 534
understanding 533
using 538
viewing results 548
Consumer System Definition Wizard 578
consuming services 533
See Also Consume Web Service wizard
prerequisites 536
contact information xl
contacts
specifying for nodes 367
container messages
adding 179
adding message parts to 189
generating schemas 193

Copyright © 1988-2007, Oracle. All rights reserved.

managing 189
understanding 189
cookies
handling in messages 227
receiving/returning 228
submitting in HTTP headers 134
creating integrations overview 31
cross-references xxxix
CSR
generating (BEA WebLogic) 605
generating (IBM WebSphere) 609
generating for client authentication 600
submitting to CAs for signing (BEA WebLogic) 606
submitting to CAs for signing (IBM WebSphere) 609
understanding 590
Customer Connection website xxxvi

D

data length view limit 434
databases
importing/exporting codesets between 404
understanding record fields 200
DB2
setting persistent cursors for messaging servers 42
supporting transformations 378
debugging
handler PeopleCode 503
handling common issues 504
integrations 503
managing 497
transform programs 380
delegation, reusing connector code via 777
DER 590
Destination Definition Wizard 580
digital certificates
certificate authorities 588
See Also CA
configuring integration gateways 53, 67
CSR 590
See Also CSR
digital signatures 588
See Also digital signatures
Distinguished Name (DN) 589
handling nonrepudiation messages 134
installing for certificate-based node authentication 592
installing for nonrepudiation 592
installing for SSL encryption (BEA WebLogic) 604
installing for SSL encryption (IBM WebSphere) 608
installing for SSL encryption (OAS) 612
installing for web server SSL encryption 603
obtaining signed public encryption keys 590
public/private encryption keys 589
understanding 588, 591
understanding installation 589
Digital Certificates page 592
digital signatures
digital certificates 588
See Also digital certificates
dispatchers
asynchronous publishing of publication contracts 24
asynchronous publishing of service operation instances 22
asynchronous subscription contracts 27
asynchronous subscription of instances 25
changing dispatcher status for processes 480
clean-up mode 479
dedicating 45
messaging servers 42
Publication Broker 21
See Also broker dispatchers/handlers
Publication Contractor 21
See Also publication dispatchers
setting parameters 48
Subscription Contractor 21
See Also subscription dispatchers
throttling messages 651
understanding 21, 48
understanding clean-up mode 479
viewing status 479
Distinguished Encoding Rules (DER) 590
Distinguished Name (DN) 589
DN 589
Document Object Model (DOM), *See* XML DOM
documentation

Copyright © 1988-2007, Oracle. All rights reserved.

printed xxxvi
related xxxvi
updates xxxvi
Domain Status page
activating/deactivating messaging server domains 479
changing dispatcher status for processes 480
setting domain grace periods 480
understanding 477
viewing dispatcher status 479
domains
activating messaging server domains 41, 479
deactivating messaging server domains 479
Domain Status page 477
See Also Domain Status page
modifying failover priorities 481
PeopleSoft Domain Administration menu 42
See Also PSADMIN
recovering from stalled queues 418
setting grace periods 480
setting up failover 480
splitting channels 370
splitting queues 45
understanding dedicated messaging servers 43
working with messaging server domains 477
DTD validation, enabling 73
duplicate routings, deleting 358
dynamic slaves 656

E

elements to ignore file configuration file 393
encoding strings 128
encryption
configuring integration gateways 53, 67
FTP target connector passwords 159
HTTP listening connectors 116
HTTP target connectors 119
integrationGateway.properties passwords 66
JMS error queue passwords 146
JMS error topic passwords 147
JMS message header passwords 148
JMS queue listener passwords 144
JMS target connector passwords 153
JMS topic subscriber passwords 145
obtaining signed public encryption keys 590
PeopleSoft 8.1 connectors 139
PeopleSoft 8.1 listening connectors 117
PeopleSoft 8.1 target connectors 119
PSCipher 65, 66
public/private keys 589
securing message parameters 132
simple file target connector passwords 156
submitting SOAP messages 135
userGatewayProfile.xml password 65
enterprise resource planning (ERP) connectors, *See* ERP connectors
Entrust Technologies 588
environment variables
%TransformData 380
ERP connectors
configuring 578
delivered with iWay SOAPswitch 572
documentation 573
generating WSDL 577
installing 572
J2EE adapter 572
Oracle Applications Database (OAP) adapter 572
SAP R/3 adapter 572
Siebel adapter 573
XML adapter 573
ERP Connectors Admin page 575
error handling 217
errors
asynchronous publishing of publication contracts 24
asynchronous subscription contracts 27
building error handling into listening connectors 763
building error handling into target connectors 759
capturing JMS connector errors in topics 147
deleting messages during upgrade 196
deleting queues during upgrade 203
error handling service 9
integration gateways 497
JMS listening connectors 143
JMS target connectors 154

Copyright © 1988-2007, Oracle. All rights reserved.

listening connectors 498
logging for integration gateways 72
logging for messages 73
logging methods and parameters 502
managing 497
managing gateway error logging 501
missing JMS message header properties 147
processing for inbound messages 252
public key certificates 594
queueing JMS listening connector errors 146
refreshing gateways 77
response message error codes 92, 93
running batch notifications 450
runtime transformations 77
setting allowable service failures for dispatchers 49
setting allowable service failures for handlers 51
setting message destinations in HTTP headers 132
setting up logging 500
stalled queues 418
submitting non-SOAP messages 136
submitting SOAP messages 135, 136
target connectors 497
transformations 381
transmitting inbound requests 134
understanding logging 10, 499
using PeopleCode to write to error queue 495
validating inbound messages 251
viewing for asynchronous service operations 443
events
OnRouteReceive 232
exceptions
integration gateways 498
Java 499
JMS target connectors 154
logging for integration gateways 72
standard 498
external message IDs
HTTP listening connector 125
JMS listening connector 143

F

failover
modifying priorities 481
setting up for domains 480
setting up for group domains 45
Failover Configuration page 480
fields
aliases 96
changes to field-level attributes 102
database record 200
excluding from rowset-based messages 184
message header/XML 200
File Transfer Protocol (FTP), *See* FTP
filtering
PeopleCode filtering example 395
understanding 373, 394
firewalls
communicating with third-party systems 695
integrating with Integration Broker systems 685
receiving third-party messages 700
sending messages to third-party systems 698
FTP
FTPS communication 160
servers 157
target connectors 119
See Also FTP target connectors
using ConnectorRequestURL 79
FTP Attachment Upload page 308
FTP Attachment utility 308
FTP servers 157
FTP target connectors
directory list support 161
required JAR files 158
setting compression properties 121
setting FTPS connector properties 160
setting properties 158
understanding 119, 157
working with non-XML files 106
FTPS communication 160

G

gateway manager
gateway services 9
integration gateway architecture 7
understanding 9
gateway private keys
setting up (BEA WebLogic) 607
setting up (IBM WebSphere) 610
Gateway Properties page 65

Copyright © 1988-2007, Oracle. All rights reserved.

gateway services 9
Gateways component
accessing/editing integrationGateway.properties 60, 64
building introspection into target connectors 758
configuring security 53
defining gateways 680
registering connectors 765
setting file security 156
setting target connector properties 120
understanding 54
Gateways page
configuring load balancing 658
defining integration gateways 55
loading connectors 57
generic routing
configuring hubs 690
understanding 689
GetMail target connectors 120
Getting Started With Integration Broker 1
glossary 793

H

handler PeopleCode
debugging 503
handlers
adding to service operation definitions 301
asynchronous publishing of instances 22
asynchronous publishing of publication contracts 24
asynchronous subscription contracts 27
asynchronous subscription of instances 25
debugging 504
dedicating 45
messaging servers 42
Publication Broker 21
See Also broker dispatchers/handlers
Publication Contractor 21
See Also publication handlers
setting parameters 51
Subscription Contractor 21
See Also subscription handlers
understanding 21, 51
headers
developing JNI headers 774
header section for request messages 86
header section for response messages 91
HTTP
connectors 116
See Also HTTP connectors
headers 127
See Also HTTP headers
HTTPS communication 116
See Also HTTPS communication
status codes 136
See Also HTTP status codes
using ConnectorRequestURL 79
HTTP connectors
adding nonrepudiation signatures 133
formatting messages 129
HTTP status codes 136
See Also HTTP status codes
listening connectors 116
See Also HTTP listening connectors
passing parameters to 130
responding to requests 134
routing transactions through proxy servers 137
submitting messages inside XML wrappers 129
submitting non-XML messages 130
submitting SOAP messages 135
target connectors 119
See Also HTTP target connectors
transmitting messages 129
understanding 123
HTTP headers
passing HTTP parameters 130
setting message destinations 132
submitting cookies 134
submitting SOAP messages 135
understanding IBInfo data 127
HTTP listening connectors
setting message parameters 123
understanding 116, 123
using external message IDs 125
HTTP status codes
submitting non-SOAP messages 136
submitting SOAP messages 136
HTTP target connectors
default remote gateway connector 368
encoding strings 128
HTTP headers 127
See Also HTTP headers
setting compression properties 121

Copyright © 1988-2007, Oracle. All rights reserved.

setting properties 121, 127
setting the content-type property 128
understanding 119, 126
HTTPS communication
HTTP connectors 123
HTTP listening connectors 116
PeopleSoft 8.1 connectors 139
PeopleSoft 8.1 listening connectors 117
PeopleSoft 8.1 target connectors 119
hubs
configuring generic-routing 690
configuring sender-specific routing 692
integrating with Integration Broker systems 688
selecting nodes 366
understanding routing 689

I

IBInfo
accessing using PeopleCode 95
receiving messages via JMS listening connectors 143
receiving third-party messages 700
retrieving data 239
understanding 228
understanding HTTP headers 127
understanding JMS headers 151
understanding request messages 82, 86
understanding response messages 92
IBM MQSeries, *See* MQSeries
IBRequest, *See* request messages
IBResponse, *See* response messages
IBUtil 554
inbound
message transmission 206
See Also inbound message transmission
messaging 206
See Also inbound messaging
Inbound Asynchronous Post page 465
Inbound File Loader Rules page 668
inbound file loader utility
about 661
development activities 663
inbound file processing 661
inbound file processing testing 673
initiating flat file processing 672
initiating processing 671
prerequisites 667
setting up processing rules 668
Inbound File Processing page 672
inbound message transmission
understanding 206
inbound messaging
handling asynchronous 232
handling cookies 227
handling synchronous 247
invoking error processing 252
request flow through the architecture 14
responding to request messages 134
routing methods 209
setting default Jolt connect string properties 70
understanding 206
validating data 251
viewing asynchronous post statistics 465
viewing performance statistics 458
viewing synchronous service operation statistics 470
Inbound Synchronous page 470
inbound transactions
adding systems 578
creating destinations/notifications 580
generating WSDL via ERP connectors 577
inheritance, reusing connector code via 777
input message name 376
input root element 376
Integration Broker
application classes 208
configuring to handle services 277
connector SDK 751
See Also connector SDK
implementing 1
understanding xliii, 5
Integration Broker Quick Configuration page 37
integration engine
architecture 10
bypassing to send messages 78
understanding 6
integration gateway
administering 54
applying transformations 74
See Also transformations
architecture 7

Copyright © 1988-2007, Oracle. All rights reserved.

bypassing the integration engine to send messages 78
configuring 55
configuring for load balancing 657
configuring security 53
connector management service 10
See Also connectors
defining 55, 680
error handling 497
error handling service 9
gateway manager 9
See Also gateway manager
gateway services 9
Gateways component 54
general configuration 54
handling common issues 504
integrationGateway.properties file 64
See Also integrationGateway.properties
local gateway compatibility 53
logging errors/messages 499
managing 53
managing error logging 501
managing message logging 500
message validation service 9
messaging objects service 9
refreshing properties 58
remote gateways 697
See Also remote gateways
running HTTP transactions through proxy servers 137
setting security properties 67
setting up error/message logging 500
setting up logging 72
specifying for nodes 368
specifying gateway versions/class-locations 68
understanding 6
understanding exceptions 498
understanding gateway definitions 35
viewing non-English characters in log files 500
XML parsing service 9
integrationGateway.properties
accessing 60, 64
configuring security 67
connectors 69
See Also connectors
controlling routing 762
enabling performance statistics feature 463
encrypting passwords 66
refreshing properties 58
setting up logging 72, 500
specifying gateway versions/class-locations 68
transforming messages at gateway 74
understanding 64, 680
integrations
asynchronous 41
See Also asynchronous integrations
configuration scenarios 679
configuring nodes 363
debugging 503
integrating between nodes (example) 737
integrating through a firewall 685
integrating via hubs 688
integrating with Integration Broker systems 683
integrating with PeopleSoft 8.1x 702
integrating with third-party systems 695, 697
See Also remote gateways
setting up overview 31
synchronous 41
See Also synchronous integrations
understanding nodes 361
understanding setup 679
International Organization for Standardization (ISO)
language codes 100
PeopleSoft timestamp formats 102
introspection
building into target connectors 758
rloading target connectors via 57
scenario for target connector development 756
iPlanet
adding JAR files for JMS connectors 141
setting JNDIFactory class names 142
ISO
language codes 100
PeopleSoft timestamp formats 102
iWay SOAPswitch
adding destinations 580
adding roles/systems/users 578
Administration Console 572

Copyright © 1988-2007, Oracle. All rights reserved.

See Also iWay SOAPswitch Administration Console
changing password/user ID 576
components 572
creating notifications 580
creating web services 579
default password/user ID 576
documentation 573
generating WSDL 577
installing 572
logging in 575
notifications 573
operation types 573
SOAPswitch servers 572
See Also iWay SOAPswitch server
understanding 571
understanding delivered adapters 572
See Also ERP connectors
web services 573
Web Services Viewer 572
iWay SOAPswitch Administration Console
accessing 577
understanding 572
iWay SOAPswitch server
starting/stopping 575
understanding 572

J

J2EE adapter 572
JAR files
adding for JMS connectors 141
required for FTP target connectors 158
Java
exceptions 499
messaging service 149
See Also JMS
native directory interface 141
See Also JNDI
native interface 753
See Also JNI
target connectors 773
See Also Java target connectors
Xerces 773
XML DOM wrapper 770
See Also Java XML DOM wrapper
Java Messaging Service (JMS), *See* JMS
Java Native Directory Interface (JNDI), *See* JNDI
Java Native Interface (JNI), *See* JNI
Java target connectors
creating in C/C++ 774
template for creating 774
understanding 773
Java XML DOM wrapper
sample code 771
understanding 770
using wrapper classes 770
JMS
listening connectors 117
See Also AS2 listening connectors; JMS listening connectors
providers 141
See Also JMS providers
queue listeners 142, 143
setting header properties 147
See Also JMS headers
target connectors 119
See Also JMS target connectors
topic subscribers 142, 144
understanding 149
JMS connectors
JAR files 141
JMS provider support 141
listening connectors 117
See Also JMS listening connectors
setting JNDIFactory class names 142
target connectors 119
See Also JMS target connectors
understanding 141
JMS headers
setting properties for listening connectors 147
understanding IBInfo data for target connectors 151
JMS listening connectors
error handling 143
external message IDs 143
JMS headers 147
See Also JMS headers
JMS queue listeners 142
JMS topic subscribers 142
receiving messages 143
setting error queue/topic properties 146
starting/stopping 149
understanding 117, 142
JMS providers
adding generic providers 155
setting for target connectors 153
setting JNDIFactory class names 142
supported 141

Copyright © 1988-2007, Oracle. All rights reserved.

using JMS target connectors 149
See Also JMS target connectors
JMS queue listeners 142, 143
JMS target connectors
errors and exceptions 154
setting compression properties 121
setting properties 151
synchronous/asynchronous communication 150
understanding 119, 149
understanding JMS headers 151
See Also JMS headers
verifying setup 154
JMS topic subscribers 142, 144
JNDI
JMS error queues 146
JMS error topics 147
JMS queue listeners 143
JMS topic subscribers 145
setting JNDIFactory class names 142
understanding JMS connectors 141
JNI
creating headers 774
developing connectors 773
understanding 753
Jolt
setting connect gateway properties 680
setting default connect string properties 70
setting node-specific connect string properties 71
setting session pooling 74
understanding 119

K

keystore
identifying the encryption key pair 68
implementing nonrepudiation 632
setting the filepath/filename /password 68
understanding 588
understanding public/private encryption keys 589
keystores
importing signed private keys into (BEAWebLogic) 606
importing signed private keys into (IBM WebSphere) 610
Keytool utility 599

L

languages
selecting for transformations /translations 374
understanding message language codes 100
viewing non-English characters in log files 500
listening connectors
AS2 117
See Also AS2 listening connectors
building error handling/logging into 763
building servlet/non-servlet based 761
controlling message routing 761
developing 753
developing classes for 761
error handling 498
HTTP 116
See Also HTTP listening connectors
installing classes 765
integrating with third-party systems 695
invoking 761
JMS 117
See Also JMS listening connectors
message logging 501
PeopleSoft 116, 122
PeopleSoft 8.1 117, 140
receiving requests 15, 18
receiving responses 18
replacing null characters 117
understanding 8, 115
understanding delivered 116
using templates 767
load balancing 657
implementing on service operation queues 659
Local Synchronous page 474
local-to-local routing
defined 331
generating 336
regenerating 337
local-to-local routing definitions
viewing status 336
logging
application servers 503
building into listening connectors 763
building into target connectors 759

Copyright © 1988-2007, Oracle. All rights reserved.

error logging methods and parameters 502
errors and messages 10
gateway refresh errors 77
integration gateway 72
managing 497
managing for gateway messages 500
managing gateway error logging 501
message logging in connectors 500, 501
message logging methods and parameters 501
messaging errors 73
runtime transformation errors 77
setting up error/message logging 500
system messages 73
understanding error/message logging 499
viewing non-English characters 500

M

master-slave dispatchers
configuring dynamic slave dispatchers 657
configuring static slave dispatchers 657
implementing 656
MCFGetMail target connectors 120
menus
PSADMIN 42
See Also PSADMIN
Message class 218
message container message format 111
See Also message containers
understanding 111
Message Definition page 179
message definitions
adding 179
understanding 35
Message Details component
understanding/accessing 427
Message Field Properties page 184
message format
example of rowset-based message 103
fieldtype sections 97
MsgData sections 98
PeopleSoft rowset-based message format 96
rowset-based message template 97
timestamp format 102
Message Monitor component
monitoring queue status 488
pausing/testing/pinging nodes 486
setting up domain failover 480
viewing messaging performance statistics 457
message part message format 108
See Also message parts
nonrowset-based message parts 111
rowset-based message parts 108
understanding 108
message parts
adding 179
adding to container messages 189
creating 188
managing 188
understanding 188
Message Record Properties page 183
Message Schema Builder 265
selecting data in 267
viewing schema details 268
viewing schemas in 266
message schemas 315
See Also schema validation
building for rowset-based messages 269
defined 265
deleting for nonrowset-based messages 272
deleting for rowset-based messages 272
importing for nonrowset-based messages 270
modifying for nonrowset-based messages 271
schemas 185
understanding 265
validating 315
viewing 318
viewing in XML 269
viewing schema details 268
message segments 254
accessing 260
configuring nodes to handle 256
counting 258
creating 257
deleting 259
ordering for processing 258
segment numbers 257
sending 260
storing for processing 258

Copyright © 1988-2007, Oracle. All rights reserved.

storing for processing, overriding 258
understanding 255
messages 177
See Also message definitions
choosing types to use 178
conditions when they are read-only 178
container messages 189
converting characters to uppercase 178
deleting 194
deleting messages during upgrade 196
excluding fields from 184
managing 177
message definitions defined 177
message parts 188
modifying 194
non-rowset-based messages 177
nonrowset-based messages 186
restrictions for modifying 178
rowset-based messages 177, 181
single signon 364
specifying for service operations 300
types of 177
understanding 177
messages, logging system 73
MessageSegmentFromDB 258
messaging
archiving/retrieving service operation instances 453
bypassing the integration engine 78
compression 121
See Also compression
configuring messaging servers for asynchronous messaging 41
connectors 115
See Also connectors
controlling message size 238
controlling routing 761
dispatchers 21
See Also dispatchers
external message IDs 125
filtering 394
See Also filtering
gateway manager 9
See Also gateway manager
generating messages 221
generating test messages 254
handlers 21
See Also handlers
handling cookies 227
handling non-XML data 381
IBInfo data for HTTP headers 127
See Also IBInfo
identifying field-level attribute changes 102
implementing nonrepudiation 637
inbound 134
See Also inbound messaging
making working storage data available globally 382
managing logging 500
messaging services 9
modifying failover priorities 481
nodes 363
See Also nodes
nonrowset-based messages 253
See Also nonrowset-based messages
PeopleSoft rowset-based message format 96
See Also message format
prerequisites for message delivery/reception 205
processing messages 232
processing service operations in parallel 198
purging messaging tables 493
queues 197
See Also queues
receiving messages 232
replacing null characters 117
request messages 81
See Also request messages
response messages 91
See Also response messages
routing PeopleCode 209
rowset-based messages 253
See Also rowset-based messages
running batch archiving 455
sending messages 221
sending/receiving messages via PeopleCode 35
servers 41
See Also messaging servers
setting the Tuxedo queue size 52
setting up domain failover 480
setting up logging 500
submitting SOAP messages 135
throttling dispatched messages 651
transformations 74
See Also transformations
translating 399

Copyright © 1988-2007, Oracle. All rights reserved.

See Also translations
understanding logging 10, 499
understanding message delivery/reception 205
understanding process flows 205
understanding security 583
See Also security
understanding SOAP compliance 106
understanding supported message structures 81
viewing performance statistics 457
XML DOM compliance 105
See Also XML DOM
messaging format
message parts 108
messaging handlers 218
messaging part
message container 111
messaging PeopleCode*, See* PeopleCode
messaging servers
activating domains 41, 479
administrating 41
assigning queues 43
configuring 44, 48
deactivating domains 479
deleting 48
dispatchers/handlers for 42
editing queue lists 47
managing via PSADMIN 42
See Also PSADMIN
processes for asynchronous messaging 20
Publication Broker 42
See Also Publication Broker
Publication Contractor 42
See Also Publication Contractor
setting domain grace periods 480
setting persistent cursors 42
Subscription Contractor 42
See Also Subscription Contractor
understanding processes 42
using dedicated servers 43
working with dedicated servers 45
working with domains 477
metadata
backporting 675
Metadata Backport utility 675
cleaning up databases after backporting data 677
metadata backported 675
using 676
working with backported projects 677
methods
addConnectorField 758
Connect 761
CopyRowsetDelta 100
CopyRowsetDeltaOriginal 100
CopyToRowset 383
CreateNextSegment 255
CurrentSegment 255
DeleteSegment 255
ExecuteEdits 251
extracting information from the Monitor 494
getDestinationSystem() 762
GETDIRLIST 161
GetSegment 255
GetXmlDoc 246
GetXMLDoc 254
InboundPublish 250
IOnRequestSend 212
logError 502, 758
logMessage 501
notification 95
OnNotify 216
OnRequestSend 228, 229
OnRouteSend 222
OnSend 229
Ping 757
Publish 223
routing PeopleCode 209
selecting a simple file connector method for messages 157
selecting an FTP method for messages 158
selecting an HTTP method for messages 127
Send 757
SetXMLDoc 254
SyncRequest 225
UpdateSegment 255
ValidateSoapDoc 251
MIME
request messages 81
response messages 91
understanding PeopleSoft listening connectors 122
Monitor
components 414
creating custom views 491

Copyright © 1988-2007, Oracle. All rights reserved.

enabling performance statistics feature 463
extracting information from 494
filtering asynchronous service operations data 419
filtering data 415
monitoring asynchronous service operation details 427
See Also Message Details component
monitoring asynchronous service operations 416
monitoring service operation transactions 422
monitoring synchronous service operation details 438
See Also Synchronous Details component
purging messaging tables 493
running batch archiving 455
running batch error notifications 450
security 414
service operation status (asynchronous) 417
understanding 413, 414
viewing IB Info data 442
viewing monitor output 421
viewing undelivered node transactions 484
Monitor Message component
monitoring publication contracts 424
monitoring subscription contracts 425
monitoring synchronous service operation instances 435
viewing queue partitioning information 426
working with messaging server domains 477
Monitor Overview component 422
MQSeries
adding JAR files for JMS connectors 141
setting JNDIFactory class names 142
MultiChannel Framework (MCF) GetMail target connectors 120
Multipurpose Internet Mail Extension standard (MIME)*, See* MIME

N

namespaces
defined 277
New Role Definition Wizard 579
node authentication
certificate-based 648
password-based 648
Node component
Connectors page 368
Contact/Notes page 367
Properties page 367
nodes
activating 365
activating nonrepudiation 365
adding pause times 487
authenticating 364
certificates, setting up 593
configuring 363
configuring local/remote 680
copying definitions 366
debugging 504
defining properties 367
deleting definitions 370
deleting pause times 487
implementing nonrepudiation 632, 637
integrating between nodes (example) 737
overriding initial/result nodes 406
pausing/testing 486
pinging 486, 487
renaming 366, 370
representing with images 366
routing PeopleCode 209
selecting a contact manager 367
selecting codeset groups 366
selecting hub 366
selecting types 364
setting as local 365
setting connector properties 369
setting for transformations 76
setting HTTP connector properties 127
setting JMS target connector properties 151
setting Jolt connect strings 70, 71, 680
setting properties for JMS message headers 147
setting target connector properties 120
simulating receiving messages from external nodes 250
specifying connectors/gateways 368
specifying contact information 367
specifying for JMS queue listeners 144

Copyright © 1988-2007, Oracle. All rights reserved.

specifying for JMS topic subscribers 145
specifying receiving nodes 124
testing local 487
understanding 374
understanding definitions 35
understanding external 364
understanding ICType 364, 366
understanding local and remote 361
understanding PIA 364
viewing the master 366
nonrepudiation
activating for nodes 365
compressing messages 121
configuring 637
defined 584
digital certificates 588
See Also digital certificates
implementing 632
processing messages 125
signatures 133
See Also nonrepudiation signatures
understanding 632
nonrepudiation signatures
adding 133
viewing for asynchronous service operations 449
viewing for synchronous service operations 438
nonrowset-based message format 105
understanding 253
nonrowset-based messages, *See* nonrowset-based message format
adding 179
adding schemas 187
deleting schemas for 272
editing schemas 187
importing schemas for 270
managing 186
modifying schemas for 271
understanding 187
notes xxxix
notifications
creating iWay SOAPswitch notifications 580
running batch error notification processes 450
understanding iWay SOAPswitch notifications 573

O

OAP adapter 572
OAS
installing digital certificates (SSL encryption) 612
objects
ConnectorInfo 229
IBInfo 228
See Also IBInfo
Integration Broker 95
Message 253
messaging objects service 9
registering JMS-administered 151
Rowset 383
Simple Object Access Protocol (SOAP) 135
See Also SOAP
using ConnectorRequest function 78
XML parsing 9
XmlDoc 106, 383
OnRouteReceive 209
OnRouteSend 209
operations, *See* service operations
Oracle
monitoring subscription contracts 425
Oracle Applications Database (OAP) adapter 572
Oracle Applications Database (OAP) adapter 572
Oracle XSL Mapper 384
adding code to XSL maps 393
deleting record and field maps 391
Design view 388
development considerations 384
documentation 387
elements to ignore configuration file 393
installing 385
launching 385
mapping records and fields 390
modifying XSL maps 393
navigating in 388
prerequisites 384
Source view 390
specifying path to installation location 385
testing XSL maps 392
viewing XSLT code 392
outbound

Copyright © 1988-2007, Oracle. All rights reserved.

message transmission 206
See Also outbound messaging
messaging 206
See Also outbound messaging
outbound message transmission
understanding 206
outbound messaging
creating web services 579
handling asynchronous 223
handling cookies 227
handling synchronous 225
identifying SOAP faults 224
message class outbound PeopleCode 222
overriding synchronous timeout interval 227
routing methods 209
sending non-XML data 222
testing 222
transforming messages 378
understanding 206, 222
understanding delivery order 222
viewing performance statistics 458
viewing synchronous service operation statistics 472
Outbound Synchronous page 472
outbound transactions
adding back-end systems 578
generating WSDL via ERP connectors 577
overriding connectors 369
overriding gateway selections 369
output message name 376
output root element 376

P

pages
Inbound File Loader Rules page 668
inbound file processing 672
partitioning
enabling/disabling fields 198
selecting fields 200
setting partitioning subchannels 125, 148
understanding 199
understanding blocked queues 418
viewing information 426
Password Encryption Utility 66
passwords
accessing integrationGateway.properties 65
authenticating 364
encrypting for target connectors 121
iWay SOAPswitch 576
running HTTP transactions through proxy servers 137
setting for application server node 71
setting for FTP target connectors 159
setting for HTTP messages 124
setting for integration gateway 68
setting for iWay SOAPswitch 576
setting for JMS error queues 146
setting for JMS error topics 147
setting for JMS message headers 148
setting for JMS queue listeners 144
setting for JMS target connectors 153
setting for JMS topic subscribers 145
setting for keystores 68
setting for proxy authentication 128
setting for simple file target connectors 156, 157
pause times
adding to nodes 487
deleting from nodes 487
PEM
certificate signing requests (CSR) 590
See Also CSR
exporting/converting certificates 597
PeopleBooks
ordering xxxvi
PeopleCode
accessing message data 380
built-in functions 78
See Also built-in functions
classes 96
See Also classes
defining PSCAMA records 99
error handling 217
filtering 374
generating to terminate transformations 412
identifying field-level attribute changes 102
making working storage data available globally 382
manipulating message format 129
Message class 218
methods 100
See Also methods

Copyright © 1988-2007, Oracle. All rights reserved.

routing 209
sending and receiving 207
sending/receiving 35
setting target connector properties 228
simulating receiving messages from external nodes 250
SOAPDoc class 218
translation example 410
understanding message filtering 395
XMLDoc class 218
PeopleCode built-in functions, *See* built-in functions
PeopleCode, typographical conventions xxxviii
PeopleSoft 8.1
connectors 117
See Also PeopleSoft 8.1 connectors
integrating 702
listening connectors 117, 140
target connectors 119
See Also PeopleSoft 8.1 target connectors
PeopleSoft 8.1 connectors
listening connectors 117, 140
target connectors 119
See Also PeopleSoft 8.1 target connectors
understanding 139
PeopleSoft 8.1 listening connectors 117, 140
PeopleSoft 8.1 target connectors
duplicate message exception 498
setting properties 140
understanding 119, 140
PeopleSoft common application message attributes (PSCAMA), *See* PSCAMA
PeopleSoft connectors
listening connectors 116, 122
target connectors 119, 122
understanding 122
PeopleSoft Integration Broker, *See* Integration Broker
PeopleSoft listening connectors 116, 122
PeopleSoft rowset-based message format 96
PeopleSoft Server Administration utility (PSADMIN), *See* PSADMIN
PeopleSoft target connectors 119, 122
performance issues
applying transformations at integration gateway 74
archiving service operations 455
editing messaging server queue lists 47
editing XSLT transformations 75
load balancing 657
setting the Tuxedo queue size 52
throttling dispatched messages 651
understanding dedicated messaging servers 43
viewing messaging statistics 457
permissions to service operations, setting 304
persistent cursors, setting 42
pinging
dispatchers 49
external systems 755
nodes 486
Ping method 757
publication contracts 24
remote nodes 487
target nodes 124
PKI 632
planning
architecture 1
integrations 2
security 3
staff skills required 3
support 3
point-to-point routing
defined 331
point-to-point routings, creating 338
prerequisites xxxv
printed documentation xxxvi
Privacy Enhanced Mail (PEM), *See* PEM
private keys 605
See Also gateway private keys
generating (BEA WebLogic) 605
generating (IBM WebSphere) 609
generating private and public key pairs for client authentication 599
generating private and public key pairs for WS-Security 599
implementing nonrepudiation 632
importing signed keys into keystores (BEA WebLogic) 606
importing signed keys into keystores (IBM WebSphere) 610
setting distinguished name values 589
understanding encryption keys 589

Copyright © 1988-2007, Oracle. All rights reserved.

properties
SegmentCount 256
SegmentsByDatabase 256
SegmentsUnOrder 256
Provide Web Service wizard
accessing generated WSDL documents 529
complex type tags 508
features 507
locations for publishing WSDL documents 508
multiroot element 508
nonrowset-based message schema requirements 508
operation types supported 508
PartnerLinkType support 518
prerequisites 521
provided WSDL document sections 509
selecting service operations 524
selecting services to provide 524
specifying publishing options 526
target namespace 508
UDDI repositories and endpoints 509
understanding 507
understanding using 522
viewing the WSDL generation log 528
viewing WSDL documents 525
WSDL document versioning 520
WSDL URL formats 509
providing services 507
See Also Provide Web Service wizard
proxy servers
routing HTTP transactions 137
routing through 127
setting properties 137
PS_FILEDIR, setting for consuming WSDL from files 537
PSADMIN
configuring messaging servers 48
creating/assigning dedicated messaging servers 45
deleting messaging servers 48
editing messaging server queue lists 47
setting compression 94
setting the Tuxedo queue size 52
throttling dispatched messages 651
understanding 42
PSCAMA
defining records 99
understanding language codes 100
understanding message XML fields 201
understanding MsgData sections for messages 98
using audit action codes 100
PSCipher 65, 66
public key certificates
accessing properties 596
obtaining signed 590
resolving root certificate mismatches 595
understanding errors 594
public key infrastructure (PKI) 632
public keys
CSR 590
See Also CSR
digital certificates 588
See Also digital certificates
generating (BEA WebLogic) 604
generating (IBM WebSphere) 608
generating private and public key pairs for client authentication 599
generating private and public key pairs for WS-Security 599
importing (BEA WebLogic) 604
importing (IBM WebSphere) 608
infrastructure (PKI) 632
nonrepudiation 632
See Also nonrepudiation
obtaining signed certificates 590
See Also public key certificates
root certificates 590
See Also root certificates
understanding encryption keys 589
publication
asynchronous service operation publication 22
dispatchers 21
See Also publication dispatchers
handlers 21
See Also publication handlers
Publication Broker 19
See Also Publication Broker
Publication Contractor 19
See Also Publication Contractor
publication contracts 19
See Also publication contracts
synchronous service operation publication 28

Copyright © 1988-2007, Oracle. All rights reserved.

understanding blocked queues 418
Publication Broker
dispatchers/handlers 42
server processes 21
understanding 19, 22
Publication Contract Handler page 467
Publication Contractor
asynchronous service operation publication 22
dispatchers/handlers 42
server processes 21
understanding 19
publication contracts
asynchronous publishing 24
monitoring 424
understanding 19
viewing information 432
viewing statistics 467
publication dispatchers
asynchronous publishing of instances 22
asynchronous publishing of publication contracts 24
server process 21
publication dispatchers/handlers
understanding 458
publication handlers
asynchronous publishing of publication contracts 24
server process 21
viewing statistics 467

Q

queries
introspection 758
See Also introspection
transmitting URL query strings 132
Queue Definitions page 197
queues 197
adding 197
archiving service operations 198
assigning messaging servers 43, 45
deleting 201
deleting during upgrade 203
editing messaging server queue lists 47
JMS queue listeners 142
partitioning 199
See Also partitioning
processing service operations in parallel 198
Queue Status page 488
refreshing messaging queues 50
renaming 201
selecting status 199
setting dispatcher properties 49
setting error queue properties 146
setting JMS target connectors timeouts 154
understanding 35, 197
understanding blocked 418
understanding messaging queues 19
understanding stalled 418

R

record fields
aliases 182
creating 181
records
aliases 182
changing underlying definitions 178
creating 181
defining PSCAMA records 99
deleting from rowset-based messages 184
inserting to rowset-based message definitions 182
preserving aliases 383
understanding aliases 96
understanding message record structure 178
recycle count
setting for dispatchers 49
setting for handlers 51
refreshing
gateway properties 58
gateway refresh errors 77
messaging queues 50
registration
DLLs for native library 774
JMS-administered objects 151
target connectors 765
related documentation xxxvi
remote certificates
importing/exporting 595
setting up 595
remote gateways
changing the default connector setting 680
configuring nodes 680

Copyright © 1988-2007, Oracle. All rights reserved.

integrating with Integration Broker systems 685
integrating with third-party systems 697
receiving third-party messages 699
sending messages to third-party systems 698
specifying for nodes 368
request messages
content section 82, 91
exceptions 499
header section 82, 86
IBInfo section 82, 86
incoming flow through the architecture 14
internal format 81
transforming 378
response messages
catalog entries 93
content section 93
error codes 93
exceptions 499
header section 91
IBInfo section 92
internal format 91
root certificates
accessing certificate properties 596
exporting/converting 597
importing signed certificates 601
installing 592
installing application server-based digital certificates 592
obtaining signed certificates 601
resolving mismatches 595
understanding 590
root records 181
routing
controlling 761
default connector 680
generic 689
See Also generic routing
PeopleCode 209
Publication Broker 20
sender-specified 689
See Also sender-specified routing
routing actions upon save 336
routing definition
overriding connector properties 346
routing definitions 332, 333
See Also routing introspection; system-generated routing definitions; user-defined routing definitions
activating 354
activating from service operation definitions 304
adding to service operation definitions 303
creating 338
defined 331
defining 332
deleting 356
deleting duplicates 358
deleting/renaming transform programs 380
generated during node introspection 333
generating, summary 333
inactivating 354
inactivating from service operation definitions 304
methods for generating 332
naming conventions 333
overriding integration gateway 346
overview 331
renaming 356
system-generated at runtime 332
system-generated during consuming services 332
system-generated during upgrade 332
types 331
user-defined 333
routing introspection
prerequisites 348
selecting nodes to introspect 350
selecting routing definitions to generate 351
selecting service operations for 348
understanding 347
viewing introspection results 353
routing methods
OnAckReceive 214
OnRequest 215
routing parameters 338
routing status 336
routings, *See* routing definitions
rowset-based message format 96
understanding 253
rowset-based messages
adding 179

Copyright © 1988-2007, Oracle. All rights reserved.

building schemas for 269
deleting records from 184
deleting schemas 186
deleting schemas for 272
field name aliases 185
inserting records 182
managing 181
record structure 178
root records 181
specifying record aliases 184
underlying record definitions 178
understanding 181
Run Archive page 455
runtime schema validation, enabling 319

S

SAP R/3 adapter 572
schema namespace, setting 279
Schema page 187
schema validation
enabling runtime schema validation 319
prerequisites 315
schemas
adding to nonrowset-based messages 187
deleting for rowset-based messages 186
editing for nonrowset-based messages 187
generating for container messages 193
generating for rowset-based messages 185
restrictions 102
Secure Sockets Layer (SSL), *See* SSL
security
adding roles 579
AS2 target connectors 119
certificate authorities (CA) 588
See Also CA
configuring integration gateways 67
digital certificates 588
See Also digital certificates
digital signatures 648
See Also digital signatures
Distinguished Name (DN) 589
FTP target connectors 119
HTTP listening connectors 116
HTTP target connectors 119
nonrepudiation 133
See Also nonrepudiation
PeopleSoft 8.1 connectors 139
planning for 3
running batch error notifications 450
sending message parameters 132
setting properties 67
setting properties for FTPS communication 160
simple files 156
submitting SOAP messages 135
transmitting URL query strings 132
understanding 583
segment batch processing
cleaning up orphaned data 490
Segment Data Cleanup page 490
SegmentCount 258
SegmentsByDatabase 258
SegmentsUnOrder 258
sender-specified routing
configuring hubs 692
understanding 689
sending and receiving
application classes 208
PeopleCode 207
servers
FTP 157
iWay SOAPswitch 572
See Also iWay SOAPswitch server
messaging 41
See Also messaging servers
proxy 137
See Also proxy servers
Service Configuration page 194, 279
service definitions
accessing 282
adding 284
configuring 285
deleting 289
renaming 289
viewing 282
viewing component-interface-based service definitions 328
service namespace, setting 279
service operation
accessing 293
viewing 295
service operation definitions 291
See Also service operation versions
adding 298
adding handlers 301

Copyright © 1988-2007, Oracle. All rights reserved.

adding routing definitions 303
attachment information, processing 310
attachment information, sending 309
configuring 298
defining versions 300
deleting 311
renaming 311
specifying fault messages 301
specifying messages 300
uploading attachments 308
service operation mapping 334
service operation queues, *See* queues
service operation types
Asynchronous – One Way 292
Asynchronous Request/Response 292
Asynchronous to Synchronous 292
defined 291
overview 12
Synchronous 292
service operation versions
using non-default versions 307
service operations 291
See Also service operation definitions; service operation versions
aliases 292
choosing component interface methods to include in 324
defined 291
monitoring information 413
See Also Monitor
resubmitting/canceling 440
service operation types 291
setting permissions 304
service operations definitions
changing services associated to 305
Service Operations Monitor, *See* Monitor
Monitor 413
See Also Monitor
service operations versions
creating 307
service system status
defined 277
development mode defined 277
production mode defined 277
setting 279
services 275, 507, 533
See Also consuming service; providing services; service definitions
configuring Integration Broker to handle 277
consuming 533
generating component interface-based services 327
restricting access to 287
selecting component interfaces to expose as 323
understanding 275
viewing WSDL documents for 283
Services page 282
services, gateway, *See* gateway services
session pooling, setting 74
setting up integrations overview 31
Siebel adapter 573
signatures
nonrepudiation 133
See Also nonrepudiation signatures
using line feeds 134
simple file target connectors
setting compression properties 121
setting file-security/properties 156
understanding 120, 156
Simple Mail Transfer Protocol (SMTP) target connectors, *See* SMTP target connectors
SMTP target connectors
setting compression properties 121
setting properties 174
understanding 120, 174
SOAP
defining PSCAMA records 99
enabling access for third-party systems 128
identifying faults 224
iWay SOAPswitch 571
See Also iWay SOAPswitch
setting message destinations in HTTP headers 132
status codes for SOAP messages 136
submitting SOAP messages 135
understanding error codes 93
understanding SOAP-compliant messages 106
using HTTP target connectors 126
SOAPDoc class 218
SOAPUpContent property 127
Solaris 425
SSL
configuring integration gateways 53

Copyright © 1988-2007, Oracle. All rights reserved.

encrypting integration gateways 67
encrypting message parameters 132
HTTP listening connectors 116
HTTP target connectors 119
PeopleSoft 8.1 connectors 139
PeopleSoft 8.1 listening connectors 117
PeopleSoft 8.1 target connectors 119
submitting SOAP messages 135
SSL encryption
implementing 616
installing digital certificates 603
installing digital certificates for 603
setting up for BEA WebLogic 607
setting up for IBM WebSphere 610
static slaves 656
Statistics page
Selecting statistics data to view 463
viewing messaging system performance statistics 457
status
changing dispatcher status for processes 480
codes for response messages 92, 93
selecting queue status 199
setting for transformations 381
viewing dispatcher status 479
viewing for queues 488
steps, inserting into transform programs 379
structured messages, *See* rowset-based messages
Sub Queue Message Queue page 426
subchannels
applying partitioning 125, 148
setting for JMS queue listeners 144
setting subchannels for JMS topic subscribers 146
subqueues
applying partitioning 199
understanding blocked queues 418
viewing partitioning information 426
subscription
asynchronous subscription of instances 25
contracts 19
See Also subscription contracts
dispatchers 21
See Also subscription dispatchers
handlers 21
See Also subscription handlers
Subscription Contractor 19
See Also Subscription Contractor
synchronous messaging 29
understanding 19
Subscription Contract Handler page 467
Subscription Contractor
dispatchers/handlers 42
server processes 21
understanding 19
subscription contracts
asynchronous subscription of service operations 25
monitoring subscription contracts 425
setting properties 433
understanding 19
understanding asynchronous 27
viewing information 433
subscription dispatchers
asynchronous subscription contracts 27
asynchronous subscription of instances 25
server process 21
subscription dispatchers/handlers
understanding 458
subscription handlers
asynchronous subscription contracts 27
server process 21
viewing statistics 467
subscription PeopleCode
asynchronous subscription contract 27
debugging 504
suggestions, submitting xl
Sun iPlanet, *See* iPlanet
support, planning for 3
Sync Message Detail page 438
synchronous
integrations 41
See Also synchronous integrations
messaging/transmission 41
See Also synchronous messaging
Synchronous Details component
accessing/understanding 438
viewing service operation details 438
synchronous integrations
configuring messaging servers 41
synchronous messaging
applying transformations at gateway 74
debugging transformations 504
handling cookies 227
monitoring instances 435

Copyright © 1988-2007, Oracle. All rights reserved.

monitoring service operation details 438
See Also Synchronous Details component
publishing 28
responding to request messages 134
routing methods 209
submitting non-SOAP messages 136
transforming messages 378
understanding compression 94
understanding subscription 29
using JMS target connectors 150
using record/field aliases 96
viewing inbound service operation statistics 470
viewing local statistics 474
viewing outbound service operation statistics 472
viewing performance statistics 457, 458
viewing service operation content in XML 440
Synchronous operation type 292
synchronous transactions
handling inbound 247
handling outbound 225
iWay SOAPswitch operations 573
See Also iWay SOAPswitch
SOAP compliance 106
System Definition Wizard, adding systems 578
system messages, logging 73
system-generated routing definitions
initiating 336
regenerating 337
understanding 335
viewing status 336

T

tables, purging messaging 493
Target Connector Interface (TCI) 497, 757
target connectors
AS2 119
See Also AS2 target connectors
building error handling/logging into 759
building introspection into 758
compression 94
creating in C/C++ 774
developing 753
developing classes for 757
development infrastructure 753
editing properties 58
encrypting passwords 66, 121
error handling 497
exceptions 498
FTP 119
See Also FTP target connectors
GetMail 120
handling introspection 756
HTTP 119
See Also HTTP target connectors
installing classes 765
Java 773
See Also Java target connectors
JMS 119
See Also JMS target connectors
loading 57
message logging 500
overriding properties at runtime 228
overriding properties using OnRequestSend 229
PeopleSoft 8.1 119
See Also PeopleSoft 8.1 target connectors
PeopleSoft target connectors 119, 122
pinging external systems 755
processing requests 15
receiving requests 18
receiving responses 17, 18
receiving third-party messages 699
registering 765
removing properties 369
selecting 680
sending messages to third-party systems 698
setting compression properties 121
setting connector properties for nodes 369
setting default Jolt connect string properties 70
setting node-specific Jolt connect string properties 71
setting properties 120, 121
setting properties at runtime 228
setting properties using OnRequestSend 229
simple file 120
See Also simple file target connectors

Copyright © 1988-2007, Oracle. All rights reserved.

SMTP 120
See Also SMTP target connectors
specifying for nodes 368
understanding 8, 117
understanding delivered 118
using templates 766
using the target connector interface 497, 757
target location, setting 279
TCI 497, 757
templates
Java target connector 774
MsgData 98
rowset-based message format, PeopleSoft 97
sample XSLT template 745
using connector templates 766
terms 793
testing
generating test messages 254
local nodes 487
nodes 486
outbound messages 222
Thawte 588
third-party
applications 116, 122
load balancing software 657
message formatting/transmission 129
products 5
systems 128
See Also third-party systems
third-party messaging
including FieldTypes sections in messages 98
integrating with third-party systems 695
receiving messages 699
sending messages to third-party systems 698
transforming messages 375
third-party systems
configuring for integration with PeopleSoft 701
integrating via remote gateways 697
integrating with 695
receiving SOAP transactions 128
timeout
setting for handlers 51
timeout interval, overriding for synchronous transactions 227
timeouts
adding pause times to nodes 487
setting for FTP target connectors 160
setting for HTTP connectors 128
setting for JMS target connectors 154
setting for PeopleSoft 8.1 connectors 141
tracing
application servers 503
managing 497
transform programs 380
transactions
configuring ERP connectors 578
viewing undelivered node transactions 484
Transform Only 376
transform programs, *See* transformations
defining 376
developing 378
developing using Oracle XSL Mapper 384
inserting actions 379
properties 376
transformation engine
sample XSLT template 745
transformations
accessing message data 380
applying at integration gateway 74
combining with translations 378
See Also translations
debugging 504
defined 35
defining transform programs 375
deleting transform programs 380
developing 75
handling non-XML data 381
including filtering 395
input message name 376
input root element 376
integrating between nodes (example) 737
integrating Integration Broker systems via hubs 688
invoking 380
making working storage data available globally 382
output message name 376
output root element 376
preserving record and field aliases 383
renaming transform programs 380

Copyright © 1988-2007, Oracle. All rights reserved.

setting properties for gateway-based 75
setting up (example) 741, 742
subscription contracts 433
terminating 412, 756
tracing transform programs 380
transforming third-party messages 375
understanding 373, 378, 397
understanding runtime errors 77
using PeopleCode 374
using TransformData class 381
using XSLT 74, 374, 397
translations
codeset repository 399
See Also codeset repositories
codesets 399
See Also codesets
combining with transformations 378
developing 400
PeopleCode example 410
searching for data to translate 366
understanding 373, 399
using the parm tag 406
using the psft_function tag 405
using the value tag 406
using XSLT 374, 405
XSLT example 407
transmission
asynchronous 150
See Also asynchronous messaging
synchronous 150
See Also synchronous messaging
transmission, message
complying with requirements 129
responses to successful/failed 134
Tuxedo
configuring queue size 52
configuring service operation queue size 45
throttling dispatched messages 651
typographical conventions xxxviii

U

UDDI Configuration page 280
UDDI interactions 106
UDDI repositories, specifying 280
understanding 384
Universal Description, Discovery, and Integration (UDDI) interactions 106
unstructured messages, *See* nonrowset-based messages
upgrade issues
assigning target connectors to nodes 704
deleting messages 196
deleting queues 203
local gateway compatibility 53
pause time considerations 486
renaming node definitions 370
URL query strings 132
URLs
debugging 504
receiving third-party messages 700
setting for HTTP connectors 121
URL query strings 132
User Details Component page 491
user-defined routing definitions
creating 338
using
Integration Broker application classes 208

V

validation
authenticating nodes 364
inbound messages 251
queue names 45
understanding message validation 9
Verisign 588
visual cues xxxix

W

W3C
adding nonrepudiation signatures 133
documentation for HTTP headers 127
XSLT 374
See Also XSLT
warnings xxxix
warnings, logging 72, 500
web server-based digital certificates
understanding 603
web servers
WebLogic 141
See Also WebLogic
web services
creating 579
iWay SOAPswitch 571, 573
See Also iWay SOAPswitch
Web Services Viewer 572
Web Services Definition Wizard 579

Copyright © 1988-2007, Oracle. All rights reserved.

WebLogic
adding JAR files for JMS connectors 141
generating CSRs 605
generating private keys 605
generating public keys 604
importing public keys 604
importing signed private keys into keystores 606
installing digital certificates (SSL encryption) 604
setting JNDIFactory class names 142
setting up gateway private keys 607
setting up SSL encryption 607
submitting CSRs to CAs for signing 606
WebSphere
generating CSRs 609
generating private keys 609
generating public keys 608
importing public keys 608
importing signed private keys into keystores 610
installing digital certificates (SSL encryption) 608
JMS provider support 141
setting up gateway private keys 610
setting up SSL encyrption 610
submitting CSRs to CAs for signing 609
World Wide Web Consortium (W3C), *See* W3C
WSDL
creating WSDL documents 579
generating via ERP connectors 577
URL formats 509
WSDL documents
consuming from BPEL processes 557
deleting 531
sources for consuming 534
viewing for services 283
WSDL Repository page 283
WSDL URL formats 509

X

Xalan 378, 773
Xerces 773
XML
adapter (iWay SOAPswitch) 573
adding nonrepudiation signatures 133
DOM-compliant messages 105
See Also XML DOM
editing for publication contracts 432
editing for subscription contracts 433
Java XML DOM wrapper 770
See Also Java XML DOM wrapper
message XML fields 201
nonrowset-based messages 253
See Also nonrowset-based messages
parsers 9
PeopleSoft XML message wrapper 129
See Also CDATA
rowset-based messages 253
See Also rowset-based messages
saving messages in XML 120
SOAP-specific errors 135
using methods 253
using record/field aliases 96
using Xalan and Xerces 773
viewing asynchronous service operations 446
viewing for publication contracts 432
viewing for subscription contracts 433
viewing synchronous service operation content 440
XML DOM
handling non-XML data 381
Java wrapper 770
See Also Java XML DOM wrapper
SOAP-compliant messages 106
understanding 105
validating messages 251
XML message schemas, *See* schemas
XML Message Viewer page 446
XMLDoc class 218
XSLT
accessing message data 380
manipulating message content 129
sample template 745
tags 405
See Also XSLT tags
transformations 74
See Also transformations
translations 374
See Also translations
XSLT tags
parm 406
psft_function 405
value 406

Copyright © 1988-2007, Oracle. All rights reserved.

Copyright © 1988-2007, Oracle. All rights reserved.