



Enterprise PeopleTools 8.45
PeopleBook: PeopleSoft
Integration Broker

Enterprise PeopleTools 8.45 PeopleBook: PeopleSoft Integration Broker
SKU PT845IBR-B 0604
Copyright © 1988-2004 PeopleSoft, Inc. All rights reserved.

All material contained in this documentation is proprietary and confidential to PeopleSoft, Inc. ("PeopleSoft"), protected by copyright laws and subject to the nondisclosure provisions of the applicable PeopleSoft agreement. No part of this documentation may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, including, but not limited to, electronic, graphic, mechanical, photocopying, recording, or otherwise without the prior written permission of PeopleSoft.

This documentation is subject to change without notice, and PeopleSoft does not warrant that the material contained in this documentation is free of errors. Any errors found in this document should be reported to PeopleSoft in writing.

The copyrighted software that accompanies this document is licensed for use only in strict accordance with the applicable license agreement which should be read carefully as it governs the terms of use of the software and this document, including the disclosure thereof.

PeopleSoft, PeopleTools, PS/nVision, PeopleCode, PeopleBooks, PeopleTalk, and Vantive are registered trademarks, and Pure Internet Architecture, Intelligent Context Manager, and The Real-Time Enterprise are trademarks of PeopleSoft, Inc. All other company and product names may be trademarks of their respective owners. The information contained herein is subject to change without notice.

Open Source Disclosure

PeopleSoft 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 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 (c) 1999-2000 The Apache Software Foundation. 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 APACHE SOFTWARE FOUNDATION 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.

OpenSSL

Copyright (c) 1998-2003 The OpenSSL Project. All rights reserved.

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.

SSLey

Copyright (c) 1995-1998 Eric Young. All rights reserved.

THIS SOFTWARE IS PROVIDED BY ERIC YOUNG "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 AUTHOR 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.

Loki Library

Copyright (c) 2001 by Andrei Alexandrescu. This code accompanies the book:

Alexandrescu, Andrei. "Modern C++ Design: Generic Programming and Design Patterns Applied". Copyright (c) 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.

Contents

General Preface

About This PeopleBook	xix
PeopleSoft Application Prerequisites.....	xix
PeopleSoft Application Fundamentals.....	xix
Related Documentation.....	xx
Obtaining Documentation Updates.....	xx
Ordering Printed Documentation.....	xx
Typographical Conventions and Visual Cues.....	xxi
Typographical Conventions.....	xxi
Visual Cues.....	xxii
Country, Region, and Industry Identifiers.....	xxii
Currency Codes.....	xxiii
Comments and Suggestions.....	xxiii
Common Elements in These PeopleBooks	xxiii

Preface

PeopleSoft Integration Broker Preface.....	xxv
PeopleSoft Integration Broker.....	xxv

Chapter 1

Getting Started with PeopleSoft Integration Broker.....	1
PeopleSoft Integration Broker Overview.....	1
Implementing PeopleSoft Integration Broker.....	1
Other Sources of Information.....	3

Chapter 2

Understanding PeopleSoft Integration Broker.....	5
Introduction to PeopleSoft Integration Broker.....	5
Integration Gateway.....	5
Integration Engine.....	6
Integration Gateway Architecture.....	6
Architecture Elements.....	7

Connectors.....	7
Gateway Manager.....	8
Gateway Services.....	8
Integration Engine Architecture.....	9
Transaction Types.....	10
Incoming and Outgoing Message Flows.....	11
Incoming Request Message Flow.....	11
Outgoing Request Message Flow.....	13
Creating and Implementing Integrations.....	15
Understanding Messaging Elements Used in Integrations.....	15
Setting Up and Creating Integrations.....	16

Chapter 3

Administering Messaging Servers for Asynchronous Messaging.....	21
Understanding Messaging Server Administration.....	21
Messaging Servers.....	21
Messaging Servers in the DB2 UDB OS/390 and z/OS Environments.....	21
Messaging Server Processes.....	22
Understanding Dedicated Messaging Servers.....	23
Considerations When Creating Dedicated Servers.....	25
Creating and Assigning Dedicated Servers.....	26
Editing Messaging Server Channel Lists.....	27
Deleting Messaging Servers.....	28
Configuring Messaging Servers.....	28
Specifying Dispatcher Parameters.....	28
Specifying Handler Parameters.....	31
Setting the BEA Tuxedo Queue Size.....	32

Chapter 4

Managing Integration Gateways.....	35
Understanding Integration Gateway Configuration.....	35
Local Gateway Compatibility.....	35
Types of Integration Gateway Configuration.....	35
The Gateways Component.....	36
Administering Integration Gateways.....	36
Defining Integration Gateways.....	36
Registering Installed Target Connectors.....	38
Refreshing Integration Gateway Properties.....	39

Editing Connector Properties.....	39
Using the integrationGateway.properties File.....	41
Accessing the integrationGateway.properties File.....	41
Encrypting Passwords.....	44
Encrypting Passwords in the PeopleSoft Pure Internet Architecture.....	44
Encrypting Passwords Using the PSCipher Java Utility.....	44
Configuring Security and General Properties.....	45
Setting Security Properties.....	45
Specifying the Gateway Version.....	46
Specifying the Gateway Class Location.....	46
Setting General Connection Properties.....	46
Setting Logging Properties.....	48
Applying Message Transformations at the Integration Gateway.....	50
Understanding Applying Message Transformations at the Integration Gateway.....	51
Developing and Implementing Gateway-Based Transformation Programs.....	51
Setting Integration Gateway Properties for Gateway-Based Transformations.....	52
Understanding Logged Errors.....	53
Bypassing Integration Engines to Send Messages.....	54
Using the ConnectorRequest Built-In Function.....	54
Using the ConnectorRequestURL Built-In Function.....	55

Chapter 5

Understanding Supported Message Structures.....	57
Integration Broker Message Structures.....	57
Internal Message Format for Request Messages.....	57
Internal Message Format for Response Messages.....	62
Local Compression.....	65
Accessing IBInfo Elements Using PeopleCode.....	65
PeopleSoft Rowset-Based Message Format.....	66
Understanding the PeopleSoft Rowset-Based Message Format.....	67
Rowset-Based Message Template.....	67
FieldTypes Section.....	68
MsgData Section.....	68
PSCAMA	70
Identifying Changes to Field-Level Attributes.....	72
PeopleSoft Timestamp Format.....	73
Rowset-Based Message Example.....	73
Nonrowset-Based Message Structures.....	75
XML DOC-Compliant Messages.....	75

SOAP-Compliant Messages.....	75
Non-XML Files.....	76

Chapter 6

Using Listening Connectors and Target Connectors.....79

Understanding Listening Connectors and Target Connectors.....	79
Listening Connectors.....	79
Target Connectors.....	81
Working With the PeopleSoft Connectors.....	85
Understanding the PeopleSoft Connectors.....	86
Using the PeopleSoft Listening Connector.....	86
Using the PeopleSoft Target Connector.....	86
Working With the HTTP Connectors.....	87
Understanding the HTTP Connectors.....	87
Using the HTTP Listening Connector.....	87
Using the HTTP Target Connector.....	90
Complying With Message Formatting and Transmission Requirements.....	92
Understanding HTTP Status Codes.....	99
Running Integration Gateways Behind Proxy Servers.....	100
Working With the PeopleSoft 8.1 Connectors.....	101
Understanding the PeopleSoft 8.1 Connectors.....	101
Using the PeopleSoft 8.1 Listening Connector.....	101
Using the PeopleSoft 8.1 Target Connector.....	101
Working With the JMS Connectors.....	102
Understanding the JMS Connectors.....	103
Specifying JNDIFactory Class Names.....	103
Using the JMS Listening Connector.....	104
Using the JMS Target Connector.....	110
Working With the Simple File Target Connector.....	115
Understanding the Simple File Target Connector.....	115
Setting File Security.....	115
Node-Level Connector Properties.....	116
Working With the FTP Target Connector.....	116
Understanding the FTP Target Connector.....	116
Prerequisites for Using the FTP Target Connector.....	117
Specifying Required JAR Files.....	117
Setting Node-Level FTP Connector Properties.....	117
Setting Node-Level FTPS Connector Properties.....	119
Using Directory Lists.....	119

Example.....	120
Working With the SMTP Target Connector.....	122
Working With the POP3 Target Connector.....	123
Understanding the POP3 Target Connector.....	123
Working With the Default POP3 Node Definition.....	127
Passing Connector Properties as Defined.....	130
Overriding the Defined Connector Properties.....	135

Chapter 7

Defining Message Channels and Messages.....	137
Understanding Messages.....	137
Understanding Message Channels.....	138
Defining Message Channels.....	138
Configuring Message Channels.....	139
Applying Channel Partitioning.....	140
Understanding Channel Partitioning.....	140
Selecting Partitioning Fields.....	141
Granting Security Access to Message Channels.....	142
Deleting Message Channels During Upgrade.....	143
Defining Message Definitions.....	143
Using the Message Structure View.....	144
Using the Field List View.....	145
Defining Message Definition Versions.....	145
Creating New Message Versions.....	145
Renaming Message Versions.....	146
Setting Default Message Versions.....	146
Using Record Message Definitions.....	147
Understanding Message Record Structure.....	147
Inserting Records into Message Definitions.....	147
Reorganizing Record Structure.....	148
Creating Record Aliases.....	148
Configuring Message Definition Properties.....	148
Assigning Message Definitions to Message Channels.....	150
Deleting Message Definitions During Upgrade.....	151

Chapter 8

Sending and Receiving Messages.....	153
Understanding Sending and Receiving Messages.....	153

Prerequisites for Sending and Receiving Messages.....	153
Messaging Process Flows.....	154
Understanding Integration PeopleCode.....	155
Sending and Receiving PeopleCode.....	155
Routing Events.....	156
Messaging Events.....	158
Messaging PeopleCode.....	159
Accessing Integration Events.....	159
Accessing Integration Events From the Message Structure Section.....	160
Accessing Integration Events From the Message Subscription Section.....	160
Creating Message Subscriptions.....	161
Generating and Sending Messages.....	162
Understanding Outbound Messaging.....	162
Handling Outbound Asynchronous Transactions.....	163
Handling Outbound Synchronous Transactions.....	165
Overriding Synchronous Timeout Intervals at Runtime.....	167
Handling Cookies.....	168
Setting and Overriding Target Connector Properties at Runtime.....	169
Receiving and Processing Messages.....	171
Understanding Publication IDs and Subscription PeopleCode.....	171
Handling Inbound Asynchronous Transactions.....	171
Handling Inbound Synchronous Transactions.....	184
Simulating Receiving Messages from External Nodes.....	185
Processing Inbound Errors.....	187
Validating Data.....	187
Using the Exit Built-in Function.....	188
Correcting Messaging Errors.....	189
Using Message Object Functionality With Nonrowset-Based Messages.....	191
Using the SetXMLDoc Built-In Function.....	191
Using the GetXMLDoc Built-In Function.....	192
Generating Test Messages.....	192

Chapter 9

Configuring Nodes and Transactions.....	195
Understanding Configuring Nodes and Transactions.....	195
Prerequisites.....	195
Routing Types.....	195
Configuring Nodes.....	196
Understanding Configuring Nodes.....	196

Understanding Local and Remote Nodes.....	196
Creating Node Definitions.....	196
Specifying Contact Information.....	201
Defining Node Properties.....	201
Specifying Gateways and Connectors.....	202
Renaming or Deleting Node Definitions.....	203
Configuring Transactions.....	204
Understanding Transactions.....	205
Common Elements Used in This Section.....	206
Viewing Transactions.....	206
Defining Transactions.....	207
Editing Transaction Details.....	209
Editing Transaction Message Details.....	211
Editing Transaction Connector Properties.....	212

Chapter 10

Using the Integration Point Wizard.....	215
Understanding the Integration Point Wizard.....	215
Prerequisites for Using the Integration Point Wizard.....	215
Common Elements Used in This Chapter.....	216
Creating Integrations Using the Integration Point Wizard.....	216
Understanding Creating Integrations Using the Integration Point Wizard.....	216
Specifying Nodes and Transaction Types.....	218
Specifying Sender Message Information.....	219
Specifying Transformation Information.....	220
Specifying Receiver Message Information.....	222
Specifying Asynchronous Reply Message and Transformation.....	226
Viewing and Saving Integrations.....	227
Accessing Relationships Created by the Integration Point Wizard.....	232

Chapter 11

Using the Integration Point Viewer.....	235
Understanding the Integration Point Viewer.....	235
Accessing Integrations.....	235
Common Elements Used to Access Integration Points.....	236
Accessing Integration Points.....	236
Viewing Integration Information.....	236
Common Elements Used to View Integration Point Information.....	236

Viewing Asynchronous Integration Point Information.....	237
Viewing Synchronous Integration Point Information.....	239
Viewing Asynchronous-to-Synchronous Integration Point Information.....	240
Viewing Inactive Objects.....	242
Activating Inactive Objects.....	244
Activating Nodes.....	244
Activating Transactions.....	244
Activating Messages.....	244
Activating Message Subscriptions.....	244
Activating Message Channels.....	244
Activating Relationships.....	245
Activating Transaction Modifiers.....	245

Chapter 12

Applying Filtering, Transformation and Translation.....247

Understanding Filtering, Transformation, and Translation.....	247
Third-Party Considerations.....	248
Developing Transform Programs.....	248
Defining Transform Programs.....	249
Working with Transform Programs.....	250
Accessing Message Data.....	251
Making Working Storage Data Available Globally.....	253
Preserving Record and Field Aliases.....	254
Filtering Messages.....	255
Understanding Message Filtering.....	255
PeopleCode Filtering Example.....	256
Applying Transformations.....	257
Understanding Transformation.....	257
Using XSLT for Transformation.....	258
Performing Data Translation.....	259
Understanding Data Translation.....	259
Defining Codeset Groups.....	261
Defining Codesets.....	262
Defining Codeset Values.....	263
Importing and Exporting Codesets Between Databases.....	264
Deleting Codesets.....	265
Using XSLT for Data Translation.....	265
XSLT Translation Example.....	267
PeopleCode Translation Example.....	270

Terminating Transformation Programs.....	272

Chapter 13

Administering Relationships.....	273
Understanding Relationships.....	273
When to Use Relationships.....	273
Understanding Transaction Modifiers.....	274
Determining Relationship Parameters.....	275
Selecting a Transaction Type Combination.....	275
Determining Where to Define the Relationship.....	276
Configuring Relationships.....	277
Understanding Configuring Relationships.....	277
Defining Relationships.....	277
Overriding Node Properties.....	279
Overriding Contact Information.....	279
Inactivating Relationships.....	280
Deleting Relationships.....	280
Managing Transaction Modifiers.....	281
Viewing Transaction Modifiers.....	281
Defining Transaction Modifiers.....	281
Editing Transaction Modifier Details.....	284
Applying Asynchronous-to-Synchronous Modifiers.....	287
Retaining Messages at Hub Nodes.....	288
Inactivating Transaction Modifiers.....	289
Deleting Transaction Modifiers.....	289

Chapter 14

Using Integration Broker Monitor.....	291
Understanding Integration Broker Monitor.....	291
Prerequisites.....	291
Common Elements Used in This Chapter.....	292
Integration Broker Monitor Features and Components.....	292
Brokers, Contractors and Queues.....	293
Message Server Processes.....	294
Dispatchers and Handlers.....	294
Asynchronous Message Publication.....	295
Asynchronous Message Subscription	298
Synchronous Message Publication.....	300

Synchronous Message Subscription.....	300
Message Statuses.....	301
Blocked and Stalled Channels.....	303
Monitoring Messaging System Information.....	304
Working with the Monitor Message Component.....	304
Filtering Messaging Information.....	305
Viewing Output.....	307
Monitoring System Queue Information.....	307
Monitoring Asynchronous Message Instances.....	308
Monitoring Publication Contracts.....	308
Monitoring Subscription Contracts.....	308
Monitoring Synchronous Message Instances.....	308
Resubmitting and Canceling Messages for Processing.....	309
Viewing Undelivered Node Transactions.....	309
Pausing, Testing, and Pinging Nodes.....	310
Monitoring Channel Status.....	311
Viewing Channel Partitioning Information.....	311
Working with Pub/Sub Server Domains.....	312
Setting Up Domain Failover.....	314
Viewing Messaging System Performance Statistics.....	316
Running Message Queries.....	335
Archiving Message Instances.....	336
Monitoring Asynchronous Message Details.....	336
Accessing the Message Details Component.....	337
Viewing Asynchronous Message Properties.....	337
Viewing Asynchronous Message Errors.....	341
Viewing Asynchronous Messages in XML Format.....	341
Viewing Nonrepudiation Information for Asynchronous Messages.....	341
Viewing Synchronous Message Details.....	341
Accessing the Synchronous Details Component.....	342
Viewing Synchronous Message Details.....	342
Viewing Synchronous Message Errors.....	344
Running Batch Error Notification Processes.....	344
Understanding Batch Error Notification.....	344
Prerequisites for Using Batch Error Notification	345
Creating Static Error Notification Lists.....	345
Running Batch Error Notification.....	346
Running Batch Message Archiving Processes.....	347
Purging Messaging Tables.....	348
Using the Message Monitor Component Interface	349

Chapter 15

Managing Error Handling, Logging, Tracing, and Debugging.....	351
Understanding Error Handling, Logging, Tracing and Debugging.....	351
Understanding Integration Gateway Error Handling.....	351
Target Connector Error Handling.....	351
Listening Connector Error Handling.....	352
Integration Gateway Exception Types.....	352
Managing Integration Gateway Message and Error Logging.....	353
Understanding Message and Error Logging.....	353
Setting Up Message and Error Logging.....	354
Viewing Non-English Characters in Integration Gateway Log Files.....	354
Managing Message Logging.....	354
Managing Error Logging.....	355
Managing Application Server Logging and Tracing.....	357
Debugging Integrations.....	357
Debugging Subscription PeopleCode.....	357
Handling Common Issues.....	358

Chapter 16

Creating Third-Party Integrations Using WSDL.....	361
Understanding Integrations Using WSDL.....	361
Terminology.....	362
Bindings.....	363
Understanding iWay SOAPswitch.....	363
Installing iWay SOAPswitch.....	363
Components.....	363
Delivered Adapters.....	363
Operation Types.....	364
Web Services and Notifications.....	364
MySQL Database.....	364
Jetty Servlet Server.....	365
iWay SOAPswitch and Adapter Documentation.....	365
Starting and Stopping iWay SOAPswitch.....	365
Understanding Starting and Stopping iWay SOAPswitch.....	366
Common Elements Used in This Section.....	366
Starting iWay SOAPswitch Server.....	366
Stopping iWay SOAPswitch Server.....	366
Logging In to iWay SOAPswitch.....	366
Logging In to iWay SOAPswitch.....	367

Changing the iWay SOAPswitch Login User ID and Password.....	367
Generating WSDL Using the ERP Connectors.....	368
Configuring ERP Connectors.....	369
Adding Systems.....	369
Adding Consumer Systems.....	369
Adding Roles and Users.....	370
Creating Web Services.....	370
Adding Destinations.....	371
Creating Notifications.....	372
Importing WSDL Into PeopleSoft.....	372
Importing WSDL Generated Using iWay SOAPswitch.....	372
Importing WSDL from Files and URLs.....	373
Using the WSDL Repository.....	374
Creating Service Transaction Metadata for Imported WSDL.....	375
Common Elements Used to Create Service Transaction Metadata	375
Accessing the WSDL Operation Wizard.....	376
Specifying Nodes for Services.....	376
Selecting Request and Response Messages for Services.....	378
Selecting Message Channels for Service Messages.....	380
Generating Service Transaction Metadata.....	381
Viewing Generated Service Transaction Metadata.....	382
Completing Integrations.....	383
Generating WSDL from Component Interfaces and Messages.....	383
Generating WSDL from Component Interfaces.....	384
Generating WSDL from PeopleSoft Messages.....	384

Chapter 17

Setting Up Secure Messaging Environments.....	387
Understanding Secure Messaging.....	387
Types of Messaging Security.....	387
Digital Certificates.....	387
Nonrepudiation.....	388
Working With Digital Certificates.....	389
Understanding Certificate Authorities.....	389
Understanding Certificate Installation Elements.....	390
Accessing Certificate Properties.....	391
Exporting and Converting Certificates.....	392
Implementing Authentication and Nonrepudiation.....	394
Installing Node-Based Digital Certificates.....	394

Configuring Authentication.....	399
Configuring Nonrepudiation.....	400
Setting Up SSL Encryption.....	401
Understanding Installing Digital Certificates for Server Authentication SSL.....	401
Installing Digital Certificates for Server Authentication SSL on BEA WebLogic.....	401
Installing Digital Certificates for Server Authentication SSL on IBM WebSphere.....	405
Setting Up Digital Certificates for Server Authentication on Application Servers.....	409
Installing Digital Certificates for Client Authentication SSL.....	412
Configuring SSL Encryption.....	415

Chapter 18

Tuning Messaging System Performance.....	417
Understanding Tuning Messaging System Performance.....	417
Throttling Dispatched Messages Through the Messaging System.....	417
Configuring Integration Gateways for Load Balancing.....	418
Understanding Configuring Integration Gateways for Load Balancing.....	418
Configuring Load Balancing.....	418

Appendix A

Integration Scenarios.....	421
Understanding Integration Setup.....	421
Integrating with PeopleSoft Integration Broker Systems.....	425
Understanding This Scenario.....	425
Configuring the System for This Scenario.....	426
Integrating with PeopleSoft Integration Broker Systems Through a Firewall.....	427
Understanding This Scenario.....	427
Configuring the System for This Scenario.....	429
Integrating with PeopleSoft Integration Broker Systems by Using Hubs.....	430
Understanding This Scenario.....	431
Understanding Hub Routing Types.....	432
Configuring Generic-Routing Hubs.....	432
Configuring Sender-Specified Routing Hubs.....	434
Integrating with Third-Party Systems.....	437
Understanding This Scenario.....	437
Configuring the System for This Scenario.....	438
Integrating with Third-Party Systems by Using Remote Gateways.....	439
Understanding This Scenario.....	439
Sending Messages to Third Party Systems.....	440

Receiving Messages From Third-Party Systems.....	441
Sending Messages to Third-Party Systems.....	444
Receiving Messages from Third-Party Systems.....	445
Integrating with PeopleSoft 8.1x Systems.....	448
Understanding This Scenario.....	448
Configuring the System for This Scenario.....	449

Appendix B

Using the Delivered Listening Connectors and Target Connectors.....451

Understanding Using This Appendix.....	451
Prerequisites.....	452
Setting Up Meta Data.....	452
Understanding Setting Up Meta Data.....	452
Prerequisites.....	452
Creating Message Channels, Messages, and Pages.....	452
Creating Nodes and Transactions.....	453
Setting Up Integration Gateway Logging.....	455
Example 1: Using the PeopleSoft Connectors.....	455
Understanding the PeopleSoft Connector Examples.....	455
Prerequisites.....	455
Using the PeopleSoft Target Connector.....	455
Using the PeopleSoft Listening Connector.....	458
Example 2: Using the HTTP Connectors.....	459
Prerequisites.....	459
Using the HTTP Listening Connector.....	459
Using the HTTP Target Connector.....	462
Example 3: Using the PeopleSoft 8.1 Connectors.....	464
Understanding the PeopleSoft 8.1 Connectors Examples.....	464
Setting Up Data for the PeopleSoft 8.1 Connectors Examples.....	464
Using the PeopleSoft 8.1 Target Connector.....	467
Using the PeopleSoft 8.1 Listening Connector.....	467
Example 4: Using the JMS Connectors.....	467
Understanding the JMS Connector Examples.....	467
Prerequisites.....	468
Using the JMS Target Connector.....	468
Using the JMS Listening Connector.....	469
Example 5: Using the Simple File Target Connector.....	471
Writing PeopleSoft Data to Files.....	471
Reading Data Into PeopleSoft From Files.....	472

Example 6: Using the FTP Target Connector.....	473
Prerequisites.....	473
Uploading Files to FTP Servers.....	473
Downloading Files From FTP Servers.....	474
Example 7: Using the SMTP Target Connector.....	475
Prerequisites.....	475
Sending Email Messages to SMTP Servers.....	475

Appendix C

Developing Transformations.....	477
Working with the Essential Elements of Transformations.....	477
Extracting PeopleSoft Messages.....	477
Creating Application Engine Programs for Transformation Processing.....	478
Associating Transformations with Transactions.....	479
Performing Data Translation Using Codeset Repositories.....	479
Implementing Codesets Using XSLT or PeopleCode.....	485
Associating PeopleSoft Nodes with Codeset Groups.....	490
Understanding Technologies Used for Transformations.....	490
Debugging Transformations.....	492
Using Non-XML Data.....	493
Using the TransformData Object.....	494
Filtering Messages.....	494
Creating Data to Be Used During XSLT Steps.....	495
Bypassing PeopleSoft Integration Broker to Invoke Transformations.....	498
Transformation Example: Integration Between Two PeopleSoft Nodes.....	499
PeopleSoft SCM Node Message Definition.....	499
PeopleSoft CRM Node Message Definition.....	502
Setting Up the Transformation.....	504
XSLT Template Walkthrough.....	508
Associating the Transformation to a Transaction.....	513
Understanding the Output After the First XSLT Pass.....	514
Understanding the Output After the Second Codeset Pass.....	518

Appendix D

Using the Integration Broker Connector SDK.....	523
Understanding the PeopleSoft Integration Broker Connector SDK.....	523
The PeopleSoft Integration Broker Connector SDK.....	523
SDK Contents.....	524

SDK Location.....	524
SDK API Documentation.....	524
Developing Connectors.....	525
Understanding Connector Development and Implementation.....	525
Understanding General Connector Class Development Considerations.....	528
Developing Target Connector Classes.....	529
Developing Listening Connector Classes.....	533
Installing Connector Classes.....	537
Registering Connectors.....	537
Using Connector Templates.....	538
Developing Connectors Based on DOM.....	542
Understanding the Java XML DOM Wrapper.....	542
Using Java XML DOM Wrapper Classes	542
Developing and Implementing Connectors in the C/C++ Environment.....	545
Understanding the Development Process.....	545
Creating Target Connectors for the C/C++ Environment.....	546
Reusing Connector Code.....	549
Reusing Connector Code Through Inheritance.....	549
Reusing Connector Code Through Delegation.....	549

Appendix E

ISO Country and Currency Codes.....	551
ISO Country Codes.....	551
ISO Currency Codes.....	560

Glossary of PeopleSoft Terms.....	571

Index	587

About This PeopleBook

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

This preface discusses:

- PeopleSoft application prerequisites.
- PeopleSoft application fundamentals.
- Related documentation.
- Typographical conventions and visual cues.
- Comments and suggestions.
- Common elements in PeopleBooks.

Note. PeopleBooks document only page elements that require additional explanation. If a page 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 applications are defined in this preface.

PeopleSoft 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 applications.

See *Enterprise PeopleTools 8.45 PeopleBook: Using PeopleSoft Applications*.

You might also want to complete at least one PeopleSoft introductory training course.

You should be familiar with navigating the system and adding, updating, and deleting information by using PeopleSoft windows, menus, and pages. 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 applications most effectively.

PeopleSoft Application Fundamentals

Each application PeopleBook provides implementation and processing information for your PeopleSoft database. However, additional, essential information describing the setup and design of your system appears in a companion volume of documentation called the application fundamentals PeopleBook. Each PeopleSoft product line has its own version of this documentation.

The application fundamentals PeopleBook consists of important topics that apply to many or all PeopleSoft applications across a product line. 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 this central PeopleBook. It is the starting point for fundamentals, such as setting up control tables and administering security.

Related Documentation

This section discusses how to:

- Obtain documentation updates.
- Order printed documentation.

Obtaining Documentation Updates

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

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

See Also

PeopleSoft Customer Connection, <https://www.peoplesoft.com/corp/en/login.jsp>

Ordering Printed Documentation

You can order printed, bound volumes of the complete PeopleSoft documentation that is delivered on your PeopleBooks CD-ROM. PeopleSoft makes printed documentation available for each major release shortly after the software is shipped. Customers and partners can order printed PeopleSoft documentation by using any of these methods:

- Web
- Telephone
- Email

Web

From the Documentation section of the PeopleSoft Customer Connection website, access the PeopleBooks Press website under the Ordering PeopleBooks topic. The PeopleBooks Press website is a joint venture between PeopleSoft and MMA Partners, the book print vendor. Use a credit card, money order, cashier's check, or purchase order to place your order.

Telephone

Contact MMA Partners at 877 588 2525.

Email

Send email to MMA Partners at peoplesoftpress@mmapartner.com.

See Also

PeopleSoft Customer Connection, <https://www.peoplesoft.com/corp/en/login.jsp>

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, method names, language constructs, and PeopleCode reserved words that must be included literally in the function call.
<i>Italics</i>	Indicates field values, emphasis, and PeopleSoft 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 <i>O</i> .
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 ().

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 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.

See *About These PeopleBooks*, “ISO Country and Currency Codes,” ISO Country Codes.

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.

See Appendix E, “ISO Country and Currency Codes,” ISO Currency Codes.

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 PeopleSoft reference and training materials. Please send your suggestions to:

PeopleSoft Product Documentation Manager PeopleSoft, Inc. 4460 Hacienda Drive Pleasanton, CA 94588

Or send email comments to doc@peoplesoft.com.

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

Common Elements in These 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.

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.

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).

Process Monitor

Click to access the Process List page, where you can view the status of submitted process requests.

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.

Request ID

An ID that represents a set of selection criteria for a report or process.

User ID

An ID that represents the person who generates a transaction.

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.

See Also

Enterprise PeopleTools 8.45 PeopleBook: PeopleSoft Process Scheduler

Enterprise PeopleTools 8.45 PeopleBook: Using PeopleSoft Applications

PeopleSoft Integration Broker Preface

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

PeopleSoft Integration Broker

PeopleSoft Integration Broker facilitates 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 messaging integrations. These processes include configuring messaging systems and managing integration gateways. They also include defining message channels, messages, nodes, transactions, relationships, and transaction modifiers.

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

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

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.

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 transforms the structure of messages and translates data according to specifications that you define.

Implementing PeopleSoft Integration Broker

PeopleSoft Integration Broker is installed as part of the PeopleTools installation process. Information about configuring the integration gateway, creating messages 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

Evaluate historical integration data, current data, as well as expected growth and increased traffic. 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, which should lead to architecture planning, hardware sizing, and so on.

Planning Integrations

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

- 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. Are the source or target systems capable of handling XML? Do they require a specific schema/protocol? Do language or character set requirements exist?

- Prioritize integration development.

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

- Determine which integrations will require synchronous messaging and which ones will require asynchronous messaging.
- Determine if data will need transformation or translation.

Determining Security

Consider the following security-related points:

- Evaluate if system-level authentication is required.
- Determine security requirements for specific integrations.

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.

Planning for Support

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

- Determine who in your organization should support integration development and administration.
- Determine the type of error-notification and exception handling to implement to meet your support requirements.

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 areas:

- PeopleTools.
- PeopleCode.
- Application Engine.
- Application Messaging.

In addition, developers should have an understanding and research capabilities in Extensible Markup Language (XML), Simple Object Access Protocol (SOAP) and Hypertext Transfer Protocol (HTTP).

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 tools 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 xxv

Enterprise PeopleTools 8.45 PeopleBook: Getting Started with Enterprise PeopleTools

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.
- Creating and implementing integrations.

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 facilitates synchronous and asynchronous messaging among internal systems and trading partners, while managing message structure, message format, and transport disparities. Because of its modular design, many elements that you develop for an integration using Integration Broker can be reused in other 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.

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 6

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, including PeopleSoft rowsets, XML document object model, Simple Object Access Protocol (SOAP), and nonrowset-based data.
- 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.
- 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 9

Integration Gateway Architecture

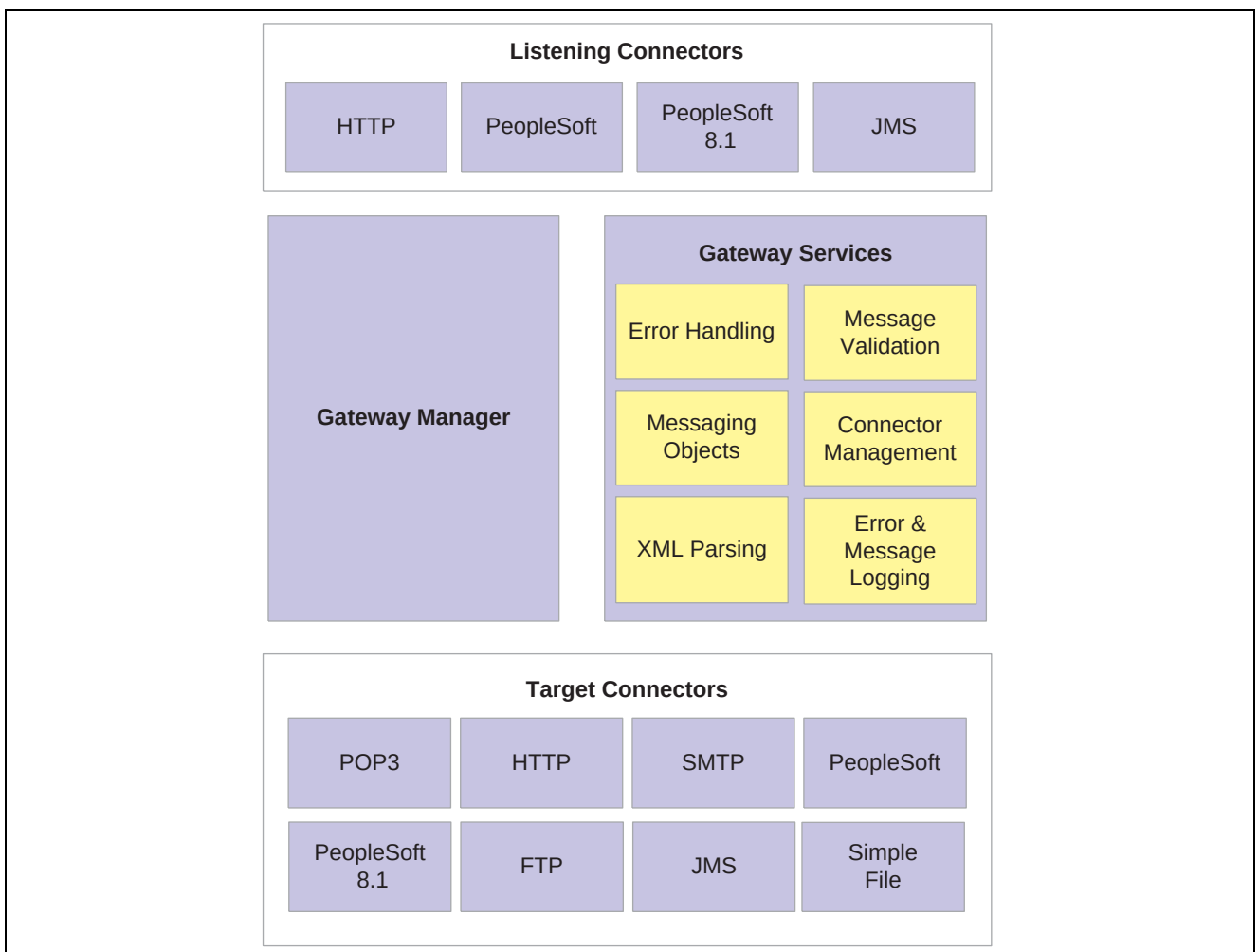
This section discusses:

- 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.



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 6, “Using Listening Connectors and Target Connectors,” page 79

Appendix D, “Using the Integration Broker Connector SDK,” page 523

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 message request. The gateway manager uses the messaging objects `IBRequest` and `IBResponse` to determine how to route each message.

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.

Messaging 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 5, “Understanding Supported Message Structures,”](#) page 57.

XML Parsing

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

Most of the time, these content sections contain XML data. Consequently, often connectors must parse and traverse XML. Many developers find that the standard Java XML objects are cumbersome for manipulating XML, so the integration gateway provides 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 8.45 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 message validation—such as making sure that the message identifies its requestor and message name—to ensure that the integration engine and the target application can process them.

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 message logs to see the IBRequest object, the IBResponse object, and even the raw data exchanged with integration participants.

See [Chapter 15, “Managing Error Handling, Logging, Tracing, and Debugging,”](#) page 351.

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.

Transaction Types

PeopleSoft Integration Broker supports four types of messaging transactions. For any transaction type, the application must invoke PeopleCode to generate and send a message, or to receive and process a message.

Transaction Type	Actions
Outbound asynchronous	1. The application generates and sends a request message. 2. One or more target system receives and processes the request message.
Outbound synchronous	1. The application generates and sends a request message. 2. The application suspends activity and waits for a response message. 3. A single target system receives and processes the request message, then generates and sends a response message. 4. The application resumes its activity and receives and processes the response message.
Inbound asynchronous	1. A source system generates and sends a request message. 2. The application receives and processes the request message.
Inbound synchronous	1. A source system generates and sends a request message. 2. The source system suspends activity and waits for a response message. 3. The application receives and processes the request message, then generates and sends a response message. 4. The source system resumes its activity and receives and processes the response message.

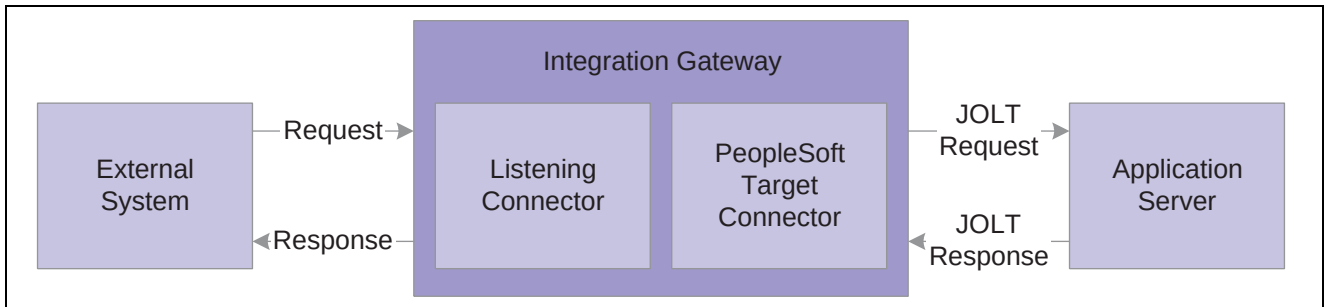
When PeopleSoft Integration Broker sends a message, the receiving system returns a response message back to the sender. With asynchronous transactions, the response is automatically generated by the integration gateway, and it serves only to notify the sending system of the transmission status of the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

Incoming and Outgoing Message Flows

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

Incoming Request Message Flow

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



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 message for use by the application server.

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

A term often used in conjunction with listening connectors is *credentials*. Incoming requests are thought to have two logical parts: the credentials and the message 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 message body. Since the credentials are separate from the body, the integration gateway does not need to parse or otherwise examine the message body for information on how to route the message.

A message without credentials cannot be processed. If the integration gateway receives such a message 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 message 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 message 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 message is received by the connector, it builds a MIME message. Typically the MIME message will only have two sections. In the first, the message credentials are stored in an XML document in a specific format. The second section stores the message body.

At this point the message is in a standard format understood by both the gateway and the application server. All messages 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.

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 message 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 message is parsed and is used to build a request object. The MIME structure is not propagated into the server.

Assuming the message parses without error, the application server must pre-process the message before the request can actually be handled.

Pre-processing involves:

- Authenticating the message, depending on the authentication scheme configured. If the request fails authentication, an error is returned.
- Determining if there is a transaction corresponding to the request in the Node Transaction table. Aside from Ping messages, all incoming messages must resolve to a single, unique entry in this table. If a suitable transaction cannot be found for a request, an error occurs.
- Determining the message handler to invoke. Currently, there are three message types supported by the integration broker: Ping, Synchronous, and Asynchronous. The message type determines the handler code to invoke. Synchronous messages are passed to sync-specific code, and asynchronous messages are passed to the publish/subscribe subsystem.

Once messages have been passed to their respective handlers, further processing is dictated by the data and PeopleCode specific to a particular system. Asynchronous requests will be published, and subscription PeopleCode will be executed. Synchronous messages may invoke PeopleCode, 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 message 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 message being exchanged. When the gateway sends a request to the application server, it expects and must get a response.

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

Asynchronous requests behave differently. Unlike sync requests, there is no blocking. A response is generated for an asynchronous request as soon as the request is written to 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 message. Receipt of such a response is not a guarantee that any applicable subscription PeopleCode has been successfully run.

Response messages 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.

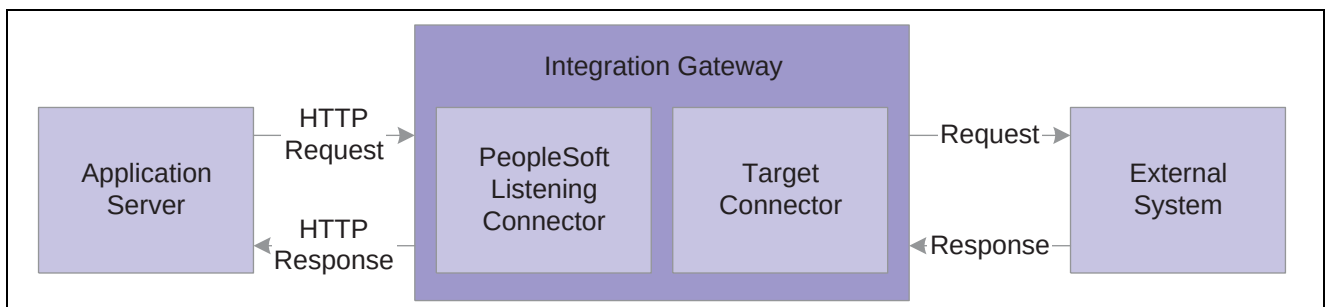
Step 7: Response is Received by the Listening Connector

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 request messages is done synchronously. A request is received by a listening connector which then converts 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 Message 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 message being sent out of the broker. Messages can be sent in PeopleCode by using the Publish or SyncRequest methods of the Message class. If the PeopleSoft Integration Broker is operated in a hub configuration, incoming synchronous messages may be set up to be routed out to other systems.

Regardless of how the message is created, the mechanism for sending the message 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 message 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 to be sent to.

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.

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 message into whatever format is required by the protocol.

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

Creating and Implementing Integrations

This section discusses:

- Messaging elements used in integrations.
- Steps for setting up integrations.

Understanding Messaging Elements Used in Integrations

You use the following elements to create and implement integrations.

Message channel definitions	A channel isolates different groups of messages from each other. Messages in the same channel are processed sequentially. Each participating node must contain a channel definition by the same name.
Message definitions	This definition holds the message data. Each participating node must contain a message definition by the same name, with the same structure. For synchronous integrations, you must provide a response message definition in addition to the request message definition.
Sending or receiving PeopleCode	PeopleCode used in integration consists of routing event that route messages to and from PeopleSoft integration broker, messaging PeopleCode that enables you to construct and manipulate data content, and publishing PeopleCode that actually publishes or sends messages.
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.
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 <i>default local node</i> 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
Transaction definitions	A transaction assembles integration meta data together to form an integration. A transaction specifies the default local node, request message definition and version, a transmission type (asynchronous or synchronous), and a direction (outbound from or inbound to the default local node). If the transaction is synchronous, it also specifies a response message definition and version. The transaction definition is part of the other participating node's definition.
Relationships and transaction modifiers	A <i>relationship</i> reconciles incompatible transaction parameters to successfully transmit data from the source to the target. Every integration requires at least one transaction that is defined at each participating PeopleSoft Integration Broker node. A source node

uses a transaction to send a message, and one or more target nodes use transactions to receive the message. If the transactions at the source and target nodes are defined with different parameters, they'll be incompatible, and the message won't be successfully transmitted.

You can apply a *transaction modifier* to the initial transaction to produce a resulting transaction that is compatible with the target node. You define a transaction modifier with respect to a specified pair of nodes, which forms a relationship.

Transformation programs

A transformation or transform program is a type of Application Engine program that you develop and specify as part of a transaction modifier. 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.

Setting Up and Creating Integrations

This section provide a high-level overview of the steps to setup and create integrations.

The Getting Started With Using PeopleSoft Integration Broker chapter in this PeopleBook also provides information for preparing to create integrations and using PeopleSoft Integration Broker.

See [Chapter 1, “Getting Started with PeopleSoft Integration Broker,” page 1](#).

Determining the Messaging Architecture

In this step you determine with 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.
- PeopleSoft Integration Broker systems by using hubs.
- Third-party systems.
- Third-party systems by using remote gateways.
- PeopleSoft 8.1x systems.

See [Appendix A, “Integration Scenarios,” page 421](#).

Installing Web Servers

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

See 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 *PeopleTools 8.45 Install Guide for your database*.

Installing Application Databases

After you install PeopleTools, install your application database.

See *PeopleTools 8.45 Install Guide for your database*.

Installing and Starting the PeopleSoft Pure Internet Architecture and Integration Gateways

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.
The location of the PeopleSoft Pure Internet Architecture setup file is <PS_HOME>\setup\mpinternet\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>\webserver\startPIA.cmd.

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

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

See *PeopleTools 8.45 Install Guide for your database*.

Installing, Configuring, and Starting Application Servers

After you install PeopleTools complete the following steps.

1. Configure an application server domain using the PSAdmin utility and boot the domain.
The location of the PSAdmin utility is <PS_HOME>\appserver\psadmin.exe.
Along with other required parameters, in the Quick-configure Menu be sure to set the Pub/Sub Servers option to Yes.
2. Boot the application server domain.
3. Verify that the application server is properly configured by starting PeopleSoft Application Designer.

See *Enterprise PeopleTools 8.45 PeopleBook: System and Server Administration*.

Activating Pub/Sub Server Domains

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

To activate pub/sub server domains, access the Domain Status page in Integration Broker Monitor. In the Domains section of the page, locate your machine name and select Active from the drop-down list and click the Refresh button.

See [Chapter 14, “Using Integration Broker Monitor,” Working with Pub/Sub Server Domains, page 312.](#)

Defining Integration Gateways and Loading Connectors

To define integration gateways and load connectors, use the Gateway component.

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

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

```
http://<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 [Chapter 4, “Managing Integration Gateways,” Administering Integration Gateways, page 36.](#)

Configuring Integration Gateway Properties Files

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 [Chapter 4, “Managing Integration Gateways,” Using the *integrationGateway.properties* File, page 41.](#)

See [Chapter 4, “Managing Integration Gateways,” Configuring Security and General Properties, page 45.](#)

Creating Integration Meta Data

The next step is to create the integration meta data. This includes creating the following:

1. Node definitions.

Use the Node Definitions component to define nodes. You must define one local default node that represents the local system. You must also define nodes that represent the systems with which you integrate. These nodes are often referred to as remote nodes.

2. Message channels.

Use PeopleSoft Application Designer to define message channels. Verify in the Message Channel properties that the status is set to Active for any message channels you create.

3. Message definitions.

Use PeopleSoft Application Designer to define messages. Verify in the Message Definition properties that the status is set to Active for any message channels you create.

4. Add Integration PeopleCode.

To write routing PeopleCode use the routing events available in the message definition in PeopleSoft Application Designer.

PeopleSoft provides three application classes you can use to create and manipulate message content. These classes are the Message class, XMLDoc class, and SOAPDoc class.

To send messages, you also use the Message class

Use Message class functionality on the Subscription event to receive messages.

5. Transactions.

Use the Node Definitions — Transaction page in the PeopleSoft Pure Internet Architecture to define transactions.

6. Relationships (Optional)

Use the Relationships component to define relationships.

7. Transformation programs. (Optional)

In PeopleSoft Application Designer, create a new App Engine Program object. Make sure to set the Program Type to Transform Only on the Advanced tab in the Program Properties dialog box.

See [Chapter 7, “Defining Message Channels and Messages,” page 137](#).

See [Chapter 8, “Sending and Receiving Messages,” page 153](#).

See [Chapter 9, “Configuring Nodes and Transactions,” page 195](#) and [Chapter 12, “Applying Filtering, Transformation and Translation,” page 247](#).

See [Chapter 13, “Administering Relationships,” page 273](#).

Granting Security Access to Message Channels

If you create new message channels, you must grant users access to them to view or edit channel information, as well as the messages contained within them. Moreover, granting security access to new message channels is required to see transaction information in the Integration Broker Monitor.

Users can view messages only in channels to which they have read access. All edit actions require write access.

To grant security access to message channels, you use permission lists in the Security component by navigating to PeopleTools, Security, Permissions & Roles, Permission Lists.

See *Enterprise PeopleTools 8.45 PeopleBook: Security Administration*, “Setting Up Permission Lists”.

CHAPTER 3

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 channel 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 activate them for the pub/sub system to become available.

See Also

[Chapter 14, “Using Integration Broker Monitor,” Working with Pub/Sub Server Domains, page 312](#)

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

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

To change the persistent cursors setting:

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

Messaging Server Processes

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

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

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

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

```

-----
PeopleSoft Domain Administration
-----
      Domain Name: QEDMODW

1) Boot this domain
2) Domain shutdown menu
3) Domain status menu
4) Configure this domain
5) TUXEDO command line <tmadmin>
6) Edit configuration/log files menu
7) Messaging Server Administration menu
q) Quit

Command to execute <1-7, q> :

```

PeopleSoft Domain Administration menu

From this menu, you can create new messaging servers, edit the channel 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 8.45 PeopleBook: System and Server Administration, “Using PSADMIN Menus”

Understanding Dedicated Messaging Servers

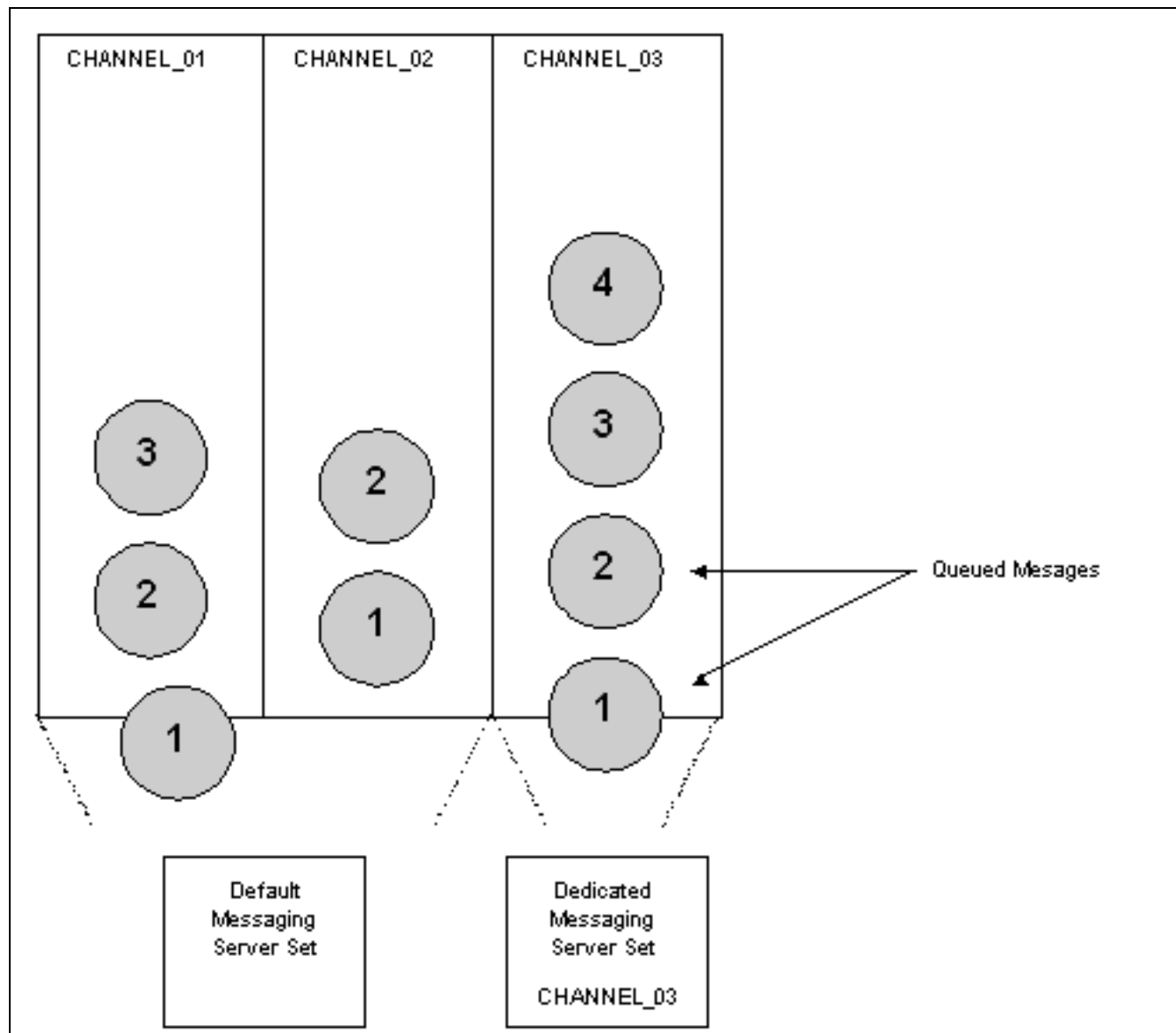
When you create a new application server domain, PSADMIN offers a set of messaging server processes that comprise the default messaging server set for that domain. The default messaging server set is sufficient for development, testing, or demonstrations.

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 channel using PSADMIN. If a given channel is the most active and creates performance bottlenecks, you can dedicate several messaging servers to that channel to cope with the message volume. A messaging server is capable of handling multiple message channels.

The following illustration depicts a dedicated messaging server set assigned to CHANNEL_03:



Dedicated messaging server set

In this scenario, the default messaging server set (_dflt process collection) continues to process the messages in the other message channels while the dedicated messaging server set processes only the messages within a specified channel. 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 channels, you must first define the message channels 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 channels.

- Configuring the new messaging server.

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

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

See [Chapter 7, “Defining Message Channels and Messages,” page 137](#).

Considerations When Creating Dedicated Servers

When creating dedicated messaging servers, consider the following points:

- There is no validation checking when you enter channel names in PSADMIN. As a result, if channel names are not spelled correctly and match those defined in the system, the dedicated server will not process any messages. Instead the default server will process them.
- Never split a channel across domains. You don't want a situation where a channel is assigned to domain A and the same channel is also assigned to domain B, since both domains will try to do the same work. You want specific channels for domain A and specific channels 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 channels already specified for other dedicated servers of the same server type. For example, do not include channel 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 3, “Administering Messaging Servers for Asynchronous Messaging,” Setting the BEA Tuxedo Queue Size, page 32](#).

- If you choose to set up group domain failover for dedicated servers, ensure that:
 - Channel sets within groups are identical.
 - Channel sets between groups are unique.

See [Chapter 14, “Using Integration Broker Monitor,” Setting Up Domain Failover, page 314](#).

- 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 channel maximum queue size.
 - Memory queue refresh rate.

See [Chapter 3, “Administering Messaging Servers for Asynchronous Messaging,” Specifying Dispatcher Parameters, page 28](#).

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 message channels.

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 channel, you can create only a dedicated publication contractor. If you haven't assigned a dedicated publication broker or a dedicated subscription contractor to the channel, the default publication broker and subscription contractor is used.

The following example shows the Message Server Administration menu:

```

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

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

    << No dedicated messaging servers are defined >>

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

Command to execute <1-3, q> : 1
  1> Publication broker
  2> Publication contractor
  3> Subscription contractor

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

```

Creating a new messaging server

To create a dedicated messaging server:

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

You can create a publication broker, a publication contractor, or a subscription contractor.

4. Enter a name to identify the new messaging server.

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

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

5. Specify the message channel that is handled by the new messaging server.

You must specify a message channel, which must already be defined in PeopleSoft Application Designer.

Note. The message channel name that you enter must exactly match the name that appears in PeopleSoft Application Designer. No prompt or validation occurs between PSADMIN and the Application Designer definitions.

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

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 Channel Lists

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

```

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

Command to execute <1-3, q> : 2

      SERVER NAME      TYPE  CHANNELS
      -----
1> PT8MSG             BRK   HRMS_01
2> PCNT01             PUB   HRMS_01
3> SCNT01             SUB   HRMS_01

Enter the number of the server to be edited: 2

Changing channel list for server 'PCNT01'...

FORMAT: Alphanumeric, max 30-char channel names separated by commas (no tabs or
spaces)
Current channels: [HRMS_01]
Enter new channels:HRMS_01,HRMS_02,HRMS_03_

```

Modifying the publication contractor channel list

To modify a channel list:

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

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

Note. The new list of message channels that you enter replaces the current list of channels for the selected messaging server. The channels that you specify must already be defined in PeopleSoft Application Designer.

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 8.45 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 Channels parameter. That parameter enables you to add message channels to the channel list. The _dflt server processes cannot be associated with a specific message channel.

Specifying Dispatcher Parameters

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

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

The following parameters apply to all three process types.

Recycle Count	Specifies the number of times each dispatcher process is executed before being terminated (intentionally) by the system and then immediately restarted. You should intermittently recycled servers to clear buffer areas. The time required to recycle a server is negligible (a matter of milliseconds), however part of the server initialization process is to rebuild the in-memory queues. The time to complete this is dependent on the number of messages residing in the pub/sub database table. The Recycle Count parameter does not translate into a native 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.
Handler Status Check Count	Determines the frequency (dispatcher cycles) that the dispatcher checks for active (booted) associated handlers. The default is 100. A dispatcher will not send any TPA calls if there are no associated active handlers. For a handler to be active, at least one process (handler) must be booted for the dispatcher to send a TPA notification.
Tuxedo Queue Status Check Count	Indicates the frequency that PeopleSoft Integration Broker reads the BEA Tuxedo queue for every cycle of the dispatcher. The higher the number, the less frequently the queue is checked. PeopleSoft recommends that you keep the default setting of 1. A value of 0 (zero) means the queue is never checked.
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 channels 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 messages published from two-tier connections. 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 that two-tier 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 two-tier publishes and error recovery.

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

Ping Rate

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

The default value is *150* seconds.

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

Maximum Ping Interval

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

Dispatcher Channel Max Queue Size

Determines the maximum number of items per channel 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 channel.

For example, with a memory queue refresh rate of *8*, multiple channels 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:

$((\text{incoming requests per second}) / (\text{number of handlers})) \times (\text{average processing time per request})$

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

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

Specifying 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

Max Retries (maximum retries)

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.

Specifies the maximum number of times that the server attempts to restart a failed action.

This parameter prevents a bad item from continuously crashing a handler process. The counter is incremented when the handler sets the status to *Working* but before it actually starts processing the item.

Setting the 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 Tuxedo 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 8.45 PeopleBook: System and Server Administration*, “Using the PSADMIN Utility”.

See *Enterprise PeopleTools 8.45 PeopleBook: System and Server Administration*, “Using PSADMIN Menus”.

CHAPTER 4

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 8.4 Integration Broker.

Types of Integration Gateway Configuration

An integration gateway requires several types of configuration:

Security configuration

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

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

See [Chapter 17, “Setting Up Secure Messaging Environments,” Setting Up SSL Encryption, page 401](#).

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 this component 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 4, “Managing Integration Gateways,” Administering Integration Gateways, page 36](#).

Administering Integration Gateways

This section discusses how to:

- Define a gateway.
- Register installed target connectors.
- Refresh the gateway properties.
- Edit available connector properties.

Defining Integration Gateways

Use the Gateway page in the Gateway component (IB_GATEWAY) to specify the gateway's location, update configuration settings, and register target connectors to be used with the gateway. To access the page, select PeopleTools, Integration Broker, 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.

Gateways

Gateway ID LOCAL

☒ Local Gateway ☐ Load Balancer

URL [Properties](#)

Connectors			Customize Find	First	1-10 of 10	Last
	*Connector ID	Description	*Connector Class Name	Properties	+	-
1	FILEOUTPUT		SimpleFileTargetConnector	Properties	+	-
2	FTPTARGET		FTPTargetConnector	Properties	+	-
3	GETMAILTARGET		GetMailTargetConnector	Properties	+	-
4	HTTPTARGET		HttpTargetConnector	Properties	+	-
5	JMSTARGET		JMSTargetConnector	Properties	+	-
6	LDAPTARGET		LDAPTargetConnector	Properties	+	-
7	POP3TARGET		POP3TargetConnector	Properties	+	-
8	PSFT81TARGET		ApplicationMessagingTargetConnector	Properties	+	-
9	PSFTTARGET		PeopleSoftTargetConnector	Properties	+	-
10	SMTPTARGET		SMTPTargetConnector	Properties	+	-

Gateways page

To define and configure a gateway:

1. Select PeopleTools, Integration Broker, 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://gatewayserver/PSIGW/PeopleSoftListeningConnector`

Gatewayserver is the machine name and port, host name, or IP address of the web server hosting the gateway. The gateway uses the PeopleSoft listening connector to receive messages from an integration engine node or a remote gateway.

4. Save the gateway definition.

See Also

[Chapter 18, “Tuning Messaging System Performance,” Configuring Integration Gateways for Load Balancing, page 418](#)

Registering Installed 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 registered and the grid is empty. You can register target connectors automatically by introspection or manually by entering information in the grid.

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

Registering 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 register it with the PeopleSoft Integration Broker connector introspection feature. Before you can register a new connector, you must install it.

See [Appendix D, “Using the Integration Broker Connector SDK,” page 523](#).

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.

Registering Connectors Manually

To register 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 register a new connector:

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 451](#)

[Appendix D, “Using the Integration Broker Connector SDK,” page 523](#)

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 Apply 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 4, “Managing Integration Gateways,” Accessing the `integrationGateway.properties` File, page 41](#).

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.

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 messages 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.

Gateways

Connector Properties

Gateway ID LOCAL

Connector SMTPTARGET

Properties						Customize	Find	First	1-8 of 8	Last
*Property ID	*Property Name	Required	Value	Default						
1 HEADER	Content-Type	☐	text/plain	☐	+ -					
2 HEADER	Content-Type	☐	text/html	☐	+ -					
3 HEADER	sendUncompressed	☒	Y	☒	+ -					
4 HEADER	sendUncompressed	☒	N	☐	+ -					
5 SMTPTARGET	BCC	☐		☐	+ -					
6 SMTPTARGET	CC	☐		☐	+ -					
7 SMTPTARGET	DestEmailAddress	☒		☒	+ -					
8 SMTPTARGET	SourceEmailAddress	☒		☒	+ -					

Properties tab for the SMTP target connector

To add a new property instance:

1. Select PeopleTools, Integration Broker, 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.
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.

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. 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
```

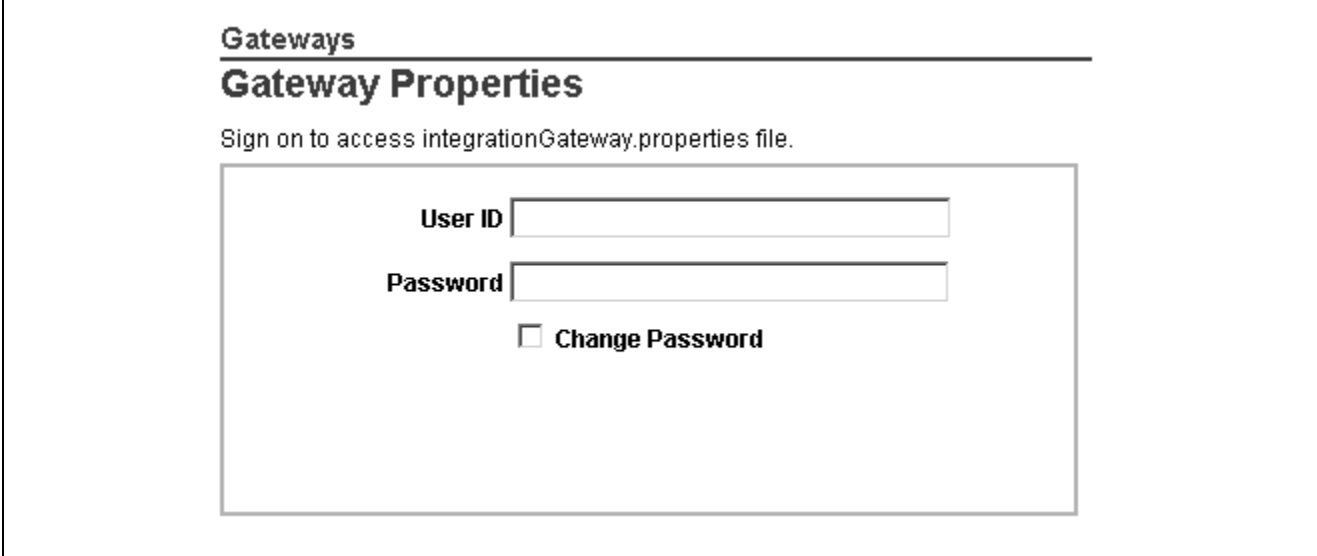
Accessing the integrationGateway.properties File

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.

Accessing the integrationGateway.properties File in the Pure Internet Architecture

The Gateway Properties page provides Pure Internet Architecture access to the integrationGateway.properties file. To access the page, select PeopleTools, Integration Broker, Gateways. Select the gateway. Click the Properties link next to the gateway URL field.

Access to the properties file in the Pure Internet Architecture is password-protected and you are prompted for a case-sensitive user ID and password each time you access the file.



Gateways

Gateway Properties

Sign on to access integrationGateway.properties file.

User ID

Password

☐ **Change Password**

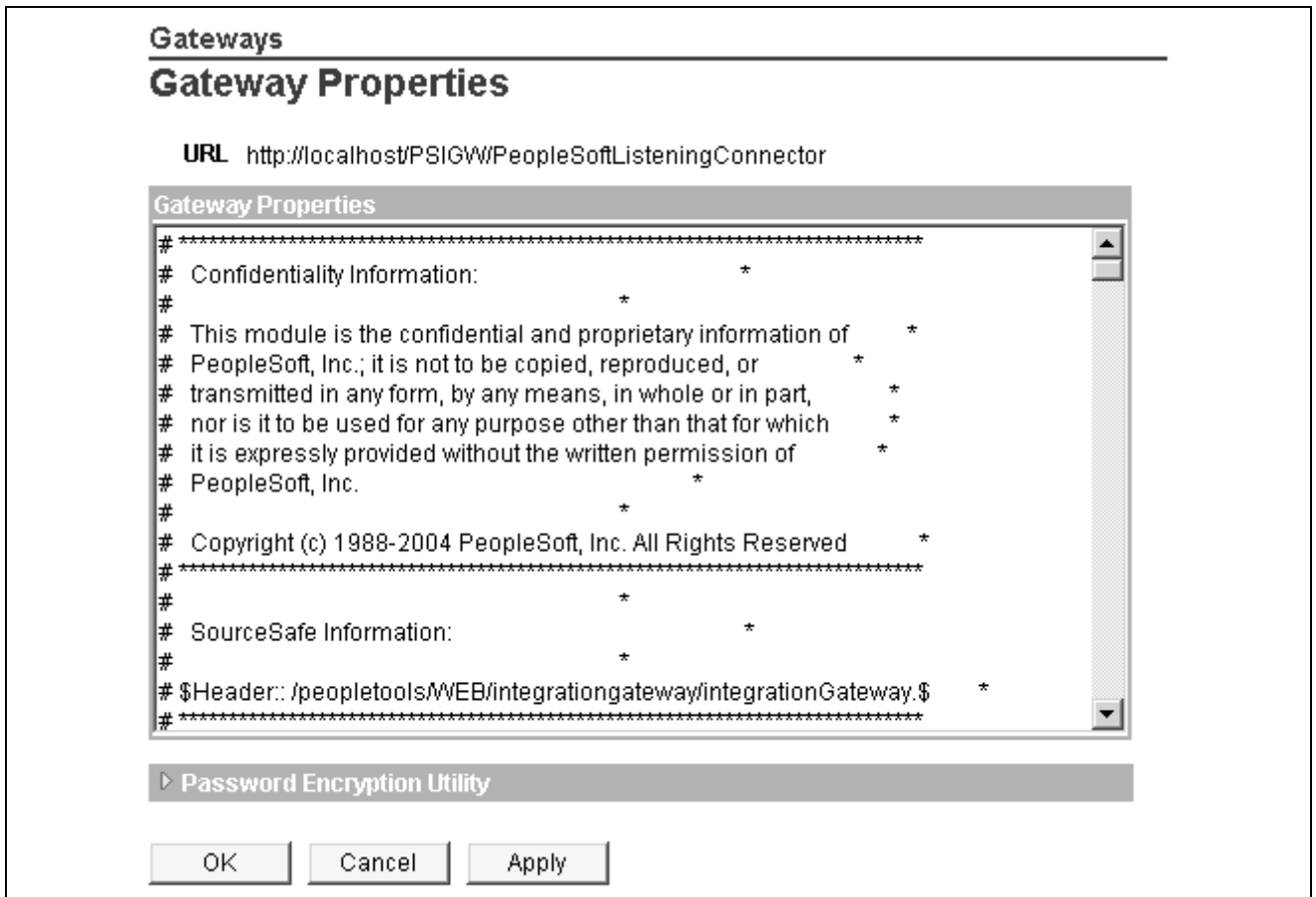
Gateway Properties sign on page in the Pure Internet Architecture

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 Gateway Properties page displays.



Gateway Properties page

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.

To access the integrationGateway.properties file in Pure Internet Architecture:

1. Select PeopleTools, Integration Broker, Gateways and select your gateway.
2. Click the Properties link next to the URL field. The Sign on to access the integrationGateway.properties file box displays.

Note. If the Gateway Properties link does not display next to the URL field, you probably have the Load Balancing URL box selected. In this case, select the Properties link for the appropriate integration gateway in the list.

3. In the User ID field, enter the user ID to access the file.
4. In the Password field, enter the password to access the file.

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 4, “Managing Integration Gateways,” Encrypting Passwords, page 44

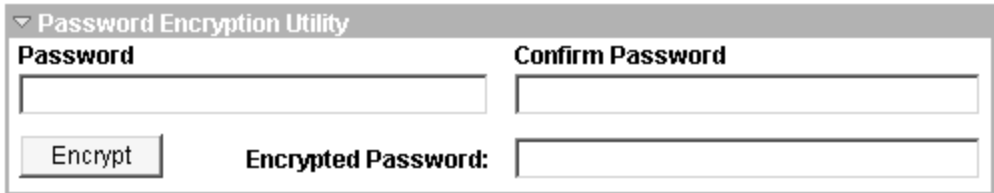
Encrypting Passwords

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>\websevr\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.

- 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.

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.

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
<code>ig.certificateAlias</code>	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.
<code>ig.certificatePasswd</code>	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 4, "Managing Integration Gateways," Encrypting Passwords, page 44 .
<code>secureFileKeystorePath</code>	Enter the full path and file name of the gateway keystore file, which is located in the web server directory structure. The path is <code><PS_HOME>\webserv\<DOMAIN>\keystore</code>
<code>secureFileKeystorePasswd</code>	Enter the keystore password, which is typically the default, <i>password</i> . This password should not be encrypted.

See Also

[Chapter 17, "Setting Up Secure Messaging Environments," page 387](#)

Specifying the Gateway Version

This required property indicates the version of PeopleTools from which the integration gateway was installed. You can find version in the section of the integrationGateway.properties file 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, 8.45.

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>\websevr\<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: <pre>com.peoplesoft.pt.integrationgateway.targetconnect⇒ or .</pre> Note. Do not change this value.

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.

To handle these cases, you can specify a default target node for the gateway if no valid target node can be determined. Use the default Jolt connect string properties:

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

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

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

```
ig.isc.serverURL=//MYMACHINE:9000
ig.isc.userid=TOPDOG
ig.isc.password=V0BN5KcQZMg
ig.isc.toolsRel=8.45
```

See [Chapter 4, "Managing Integration Gateways," Setting General Connection Properties, page 46.](#)

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: <pre>ig.isc.MYNODE.serverURL=//MYMACHINE:9000</pre> Note. You can determine the Jolt port of the application server by examining the <i>JOLT Listener</i> 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: <pre>ig.isc.MYNODE.userid=TOPDOG</pre>
ig.isc.\$NODENAME.password	Enter UserPswd that you defined when you created the application server domain. This password must be encrypted; for example: <pre>ig.isc.MYNODE.password=VOBN5KcQZMg</pre> See Chapter 4, "Managing Integration Gateways," Encrypting Passwords, page 44.
ig.isc.\$NODENAME.toolsRel	Enter the version number of PeopleTools installed on the application server node to two decimal places; for example: <pre>ig.isc.MYNODE.toolsrel=8.45</pre>

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
<code>ig.log.level</code>	Enter a numeric value to specify the desired level of gateway logging and exception handling. Values are: • <code>-100</code>: Suppresses message logging. The property is preset to this value. • <code>-1</code>: Logs language exceptions only. • <code>1</code>: Logs language and standard exceptions. • <code>2</code>: Logs all errors and warnings. • <code>3</code>: Logs errors, warnings, and important information. This is the default if you don't specify a value for this property. • <code>4</code>: Log errors, warnings, and important and standard information. • <code>5</code>: 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.
<code>ig.log.backgroundImage</code>	Specify the background JPEG image to use when displaying error and message log documents. The default location and image name <code>PSbackground.jpg</code>. By default it is located in <code><PS_HOME>\webserv\<DOMAIN>\applications\peoplesoft\PSIGW</code>. Images in the default location don't require a path, but you can specify a full path to an image file in any other location.

Message Logging Properties

Property	Description
<code>ig.messageLog.filename</code>	Enter the full path and file name of an HTML file to use as a message log. This property is preset to <code><PS_HOME>\webserv\<DOMAIN>\applications\peoplesoft\PSIGW\msgLog.html</code>.

Property	Description
ig.messageLog.maxSize	Specify the maximum size of the message log, in kilobytes (KB). This property is preset to <i>10000</i> , 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 <i>0</i> 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>\websrv\<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 <i>1000</i> , 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 <i>0</i> to retain all backed up files. This property is preset to 5.

See Also

Chapter 15, “Managing Error Handling, Logging, Tracing, and Debugging,” page 351

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 12, “Applying Filtering, Transformation and Translation,” page 247

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.

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 12, “Applying Filtering, Transformation and Translation,” Applying Transformations, page 257](#).

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 4, “Managing Integration Gateways,” Setting Integration Gateway Properties for Gateway-Based Transformations, page 52](#).

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.

Property	Description
<code>ig.isGatewayTransformationEnabled</code>	Specify whether transformation is enabled for this gateway. Valid values are: <i>TRUE</i>. Transformation is enabled. <i>FALSE</i>. Transformation is disabled — the integration gateway will ignore the other transformation properties. This is the default value.
<code>ig.DefaultServer.LocalNode</code>	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 <code>ig.DefaultServer.LocalNode=DEF_NODE</code> All transformations require that you specify both a source node and a destination node. This property applies if either the <code>ig.transformN.SourceNode</code> property or the <code>ig.transformN.DestinationNode</code> property is empty or invalid, or if the <code>IBRequest</code> doesn't specify either <code>RequestingNode</code> or <code>DestinationNode</code> .
<code>ig.transforms</code>	Specify the number of transformations configured in the <code>integrationGateway.properties</code> file; for example: <code>ig.transforms=7</code>
<code>ig.transformN.XSL</code>	Enter the full path and filename of transformation program <i>N</i> . Your path specification must use either double back slashes or single forward slashes as separators; for example <code>ig.transform4.XSL=C:\\XSLProgs\\MyTransform.xml</code> <code>ig.transform4.XSL=C:/XSLProgs/MyTransform.xml</code> <code>ig.transform4.XSL=/usr/xsls/MyTransform.xml</code>
<code>ig.transformN.MessageName</code>	Enter the name of the message to be transformed by transformation program <i>N</i> ; for example: <code>ig.transform4.MessageName=MY_MSG_A</code>

Property	Description
<code>ig.transformN.SourceNode</code>	Enter the name of the source node from which the original message is being sent, or enter the value <i>ANY</i>; for example: <pre>ig.transform4.SourceNode=NODE_Aig.transform4.SourceNode=ANY</pre> If this value is <i>ANY</i>, the value of the <code>ig.DefaultServer.LocalNode</code> property will be used instead.
<code>ig.transformN.DestinationNode</code>	Enter the name of the target node to which the transformed message is being sent, or enter the value <i>ANY</i>; for example: <pre>ig.transform4.DestinationNode=NODE_Big.transform4.DestinationNode=ANY</pre> If this value is <i>ANY</i>, the value of the <code>ig.DefaultServer.LocalNode</code> property will be used instead.
<code>ig.transformN.DestinationMessageName</code>	(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: <pre>ig.transform4.DestinationMessageName=MY_MSG_B</pre> 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.transformN.XSL`.
- No value is specified for `ig.transformN.MessageName`.
- No value is specified for both `ig.DefaultServer.LocalNode` and `ig.transformN.SourceNode`.
- No value is specified for both `ig.DefaultServer.LocalNode` and `ig.transformN.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.

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 8, “Sending and Receiving Messages,” Generating and Sending Messages, page 162](#)

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(Message.QE_FLIGHTPLAN_UNSTRUCT);

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

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

&MSG2 = ConnectorRequest(&MSG);

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


Using the ConnectorRequestURL Built-In Function

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

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

Response messages are returned in a string.

Using ConnectorRequestURL with HTTP

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

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


CHAPTER 5

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.

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:

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: <5722340.1008266844175.JavaMail.MNowose1@MNOWOSEL071900>
Date: Thu, 13 Dec 2001 10:07:24 -0800 (PST)
Mime-Version: 1.0
Content-Type: multipart/related;
    boundary="----=_Part_5_7190308.1008266838787"
Content-ID: PeopleSoft-Integration-Broker-Internal-Mime-Message

-----=_Part_5_7190308.1008266838787
Content-Type: text/plain; charset=UTF-8
Content-Transfer-Encoding: 8bit
Content-Disposition: inline
Content-ID: IBInfo

<?xml version="1.0"?><IBInfo><MessageName><![CDATA[EXAMPLE_REQUEST_MESSAGE]]>
</MessageName><MessageType>sync</MessageType><From><RequestingNode>
<![CDATA[SOURCENODE]]></RequestingNode><OrigTimeStamp>2001-11-13T10:07:18.817-0800
</OrigTimeStamp></From><PathInfo><![CDATA[/PSIGW/PeopleSoftListeningConnector]]>
</PathInfo><ContentSections><ContentSection> <ID>ContentSection0</ID>
<NonRepudiation>N</NonRepudiation></ContentSection></ContentSections></IBInfo>
-----=_Part_5_7190308.1008266838787
Content-Type: text/plain; charset=UTF-8
Content-Transfer-Encoding: 8bit
Content-Disposition: inline
Content-ID: ContentSection0

<?xml version="1.0"?>
<TestXml>This is a sample request message.</TestXml>
-----=_Part_5_7190308.1008266838787"

```

IBRequest Header Section

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

```

Message-ID: <5722340.1008266844175.JavaMail.MNowose1@MNOWOSEL071900>
Date: Thu, 13 Dec 2001 10:07:24 -0800 (PST)
Mime-Version: 1.0
Content-Type: multipart/related;
    boundary="----=_Part_5_7190308.1008266838787"
Content-ID: PeopleSoft-Integration-Broker-Internal-Mime-Message

```

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>
  <MessageName><![CDATA[EXAMPLE_REQUEST_MESSAGE]]></MessageName>
  <MessageType>sync</MessageType>
  <TransactionID><![CDATA[]]></TransactionID>
  <From>
    <RequestingNode><![CDATA[QE_LOCAL]]></RequestingNode>
    <Password><![CDATA[]]></Password>
    <DN><![CDATA[]]></DN>
    <OrigUser><![CDATA[]]></OrigUser>
    <OrigNode><![CDATA[]]></OrigNode>
    <OrigProcess><![CDATA[]]></OrigProcess>
    <OrigTimeStamp>2001-11-13T11:06:22.142-0800</OrigTimeStamp>
  </From>
  <To>
    <DestinationNode><![CDATA[TARGETNODE]]></DestinationNode>
    <FinalDestinationNode><![CDATA[]]></FinalDestinationNode>
  </To>
  <QStrArgs/>
  <Cookies><![CDATA[]]></Cookies>
  <PathInfo><![CDATA[/PSIGW/PeopleSoftListeningConnector]]></PathInfo>
  <ContentSections>
    <ContentSection>
      <ID>ContentSection0</ID>
      <NonRepudiation>N</NonRepudiation>
      <Headers>
        <version><![CDATA[VERSION_1]]></version>
      </Headers>
    </ContentSection>
  </ContentSections>
  <InternalInfo>
    <AppMsg>
      <PublishingNode><![CDATA[]]></PublishingNode>
      <Channel><![CDATA[]]></Channel>
      <SubChannel><![CDATA[]]></SubChannel>
      <DefaultDataVersion>VERSION_1</DefaultDataVersion>
      <VisitedNodes><![CDATA[|QE_LOCAL|]]></VisitedNodes>
      <PublicationID></PublicationID>
    </AppMsg>
  </InternalInfo>
  <Connector>
    <ConnectorName><![CDATA[HTTPTARGET]]></ConnectorName>
    <ConnectorClassName><![CDATA[HttpTargetConnector]]></ConnectorClassName>
    <RemoteFrameworkURL><![CDATA[]]></RemoteFrameworkURL>
    <ConnectorParameters>
      <ConnectorParam>
```

```

        <Name><![CDATA[Method]]></Name>
        <Value><![CDATA[POST]]></Value>
    </ConnectorParam>
    <ConnectorParam>
        <Name><![CDATA[URL]]></Name>
        <Value><![CDATA[http://www.externalsystem.com/serlvet]]></Value>
    </ConnectorParam>
</ConnectorParameters>
<ConnectorHeaders>
    <Header>
        <Name><![CDATA[sendUncompressed]]></Name>
        <Value><![CDATA[Y]]></Value>
    </Header>
</ConnectorHeaders>
</Connector>
</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.

The only elements that are always required are MessageName and RequestingNode.

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

Element	Description
IBInfo / MessageName	The name of the request message.
IBInfo / MessageType	(Optional) This is the type of the message. The valid values are: asynchronous, synchronous and ping.
IBInfo / TransactionID	(Optional) The transaction ID is used to uniquely identify a request.
IBInfo / From / RequestingNode	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.
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.

Element	Description
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 / InternalInfo / AppMsg / PublishingNode	(Optional) The node that published the message.
IBInfo / InternalInfo / AppMsg / Channel	(Optional) The channel to which the message was published.
IBInfo / InternalInfo / AppMsg / SubChannel	(Optional) The subchannel to which the message was published.
IBInfo / InternalInfo / AppMsg / DefaultDataVersion	(Optional) The default data version for this message on the node sending the message.
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.
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: <2351889.1008355101212.JavaMail.MNowosel@MNOWOSEL071900>
Date: Fri, 14 Dec 2001 10:38:21 -0800 (PST)
Mime-Version: 1.0
Content-Type: multipart/related;
    boundary="-----_Part_4_7210339.1008355101202"
Content-ID: PeopleSoft-Integration-Broker-Internal-Mime-Message

-----_Part_4_7210339.1008355101202
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
Content-Type: text/plain; charset=UTF-8
Content-Transfer-Encoding: 8bit
Content-Disposition: inline
Content-ID: ContentSection0

<?xml version="1.0"?>
<TestXml>This is a sample response message.</TestXml>
-----_Part_4_7210339.1008355101202--
```

IBResponse Header

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

```
<Message-ID: <2351889.1008355101212.JavaMail.MNowosel@MNOWOSEL071900>
Date: Fri, 14 Dec 2001 10:38:21 -0800 (PST)
Mime-Version: 1.0
Content-Type: multipart/related;
```



```
boundary="----=_Part_4_7210339.1008355101202"
Content-ID: PeopleSoft-Integration-Broker-Internal-Mime-Message
```

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.
IBInfo / Status / DefaultMsg	Used if the message catalog is unavailable. This value corresponds to the "Explanation" for a given entry in the message catalog.

Element	Description
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.

```
<?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.

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: <4328049.1008270392287.JavaMail.MNowosel@MNOW0SEL071900>
Date: Thu, 13 Dec 2001 11:06:32 -0800 (PST)
Mime-Version: 1.0
Content-Type: multipart/related;
    boundary="----=_Part_25_2235074.1008270392277"
Content-ID: PeopleSoft-Integration-Broker-Internal-Mime-Message

-----=_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</MsgSet>
<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.

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

See Also

Enterprise PeopleTools 8.45 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 Subscription event for a rowset-based asynchronous message.

```
Local MESSAGE &MSG;  
String &strReturn;  
  
&MSG = 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 Subscription event for a nonrowset-based asynchronous message.

```
Local MESSAGE &MSG;  
Local XmlDocument &xmldoc;  
String &strReturn;  
  
&MSG = GetMessage();  
  
For &i = 1 To &MSG.IBInfo.IBConnectorInfo.GetNumberOfConnectorProperties()  
  
    &strReturn= &MSG.IBInfo.IBConnectorInfo.GetQueryStringArgName(&i);  
    &strReturn= &MSG.IBInfo.IBConnectorInfo.GetQueryStringArgValue(&i);  
  
End-For;  
  
// To access the content data  
  
&xmldoc = &MSG.GetXmlDoc();
```

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.

Understanding the PeopleSoft Rowset-Based Message Format

To work with rowset-based messages—the PeopleSoft native format—you define a message in PeopleSoft Application Designer, 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.

Note. Rowset-based messages are also referred to as structured messages.

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 Message 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 message has multiple target nodes with different record or field naming schemes, you create for each target a separate message 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.

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 5, “Understanding Supported Message Structures,” PSCAMA , page 70](#)

[Chapter 8, “Sending and Receiving Messages,” Understanding Integration PeopleCode, page 155](#)

[Chapter 12, “Applying Filtering, Transformation and Translation,” page 247](#)

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 *fieldtype* 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 fieldtype 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"/>
    <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>
    <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 5, "Understanding Supported Message Structures," PSCAMA , page 70

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 (<i>H</i>) or trailer (<i>T</i>) or contains data (blank). The header and trailer messages don't contain 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 8.45 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:

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).

Audit Action Code	Description
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 in PeopleSoft Application Designer.

See [Chapter 8, "Sending and Receiving Messages," Generating Test Messages, page 192](#).

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"*.

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.

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"/>
  </PSCAMA>
</SDK_BUS_EXP_APPR_MSG>
```

```

    </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>

```

</SDK_BUS_EXP_APPR_MSG>

Nonrowset-Based Message Structures

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

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

Note. Nonrowset-based messages are sometimes referred to as nonrowset-based messages.

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 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 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 8, “Sending and Receiving Messages,” Generating and Sending Messages, page 162

Chapter 8, “Sending and Receiving Messages,” Receiving and Processing Messages, page 171

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.

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.

[illegible]

See Also

Chapter 6, “Using Listening Connectors and Target Connectors,” Complying With Message Formatting and Transmission Requirements, page 92

CHAPTER 6

Using Listening Connectors and Target Connectors

This chapter discusses how to:

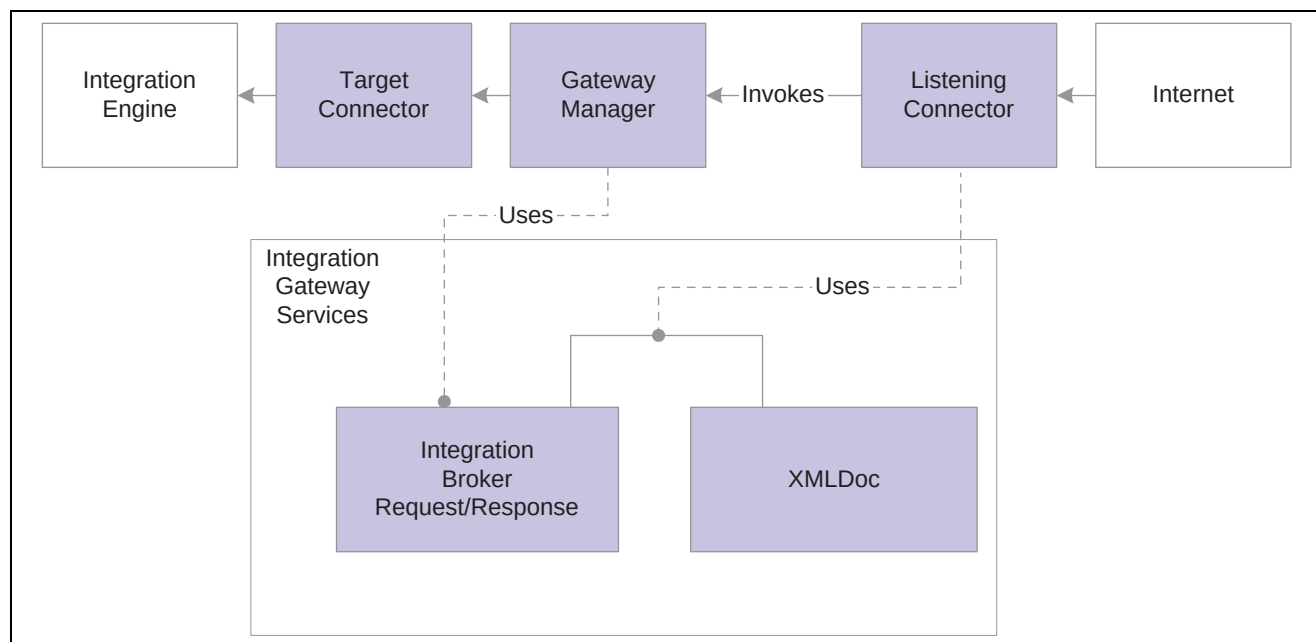
- Work with the PeopleSoft connectors.
- Work with the HTTP connectors.
- Work with the PeopleSoft 8.1 connectors.
- Work with the Java Messaging Service (JMS) connectors.
- Work with the simple file target connector.
- Work with the File Transfer Protocol (FTP) target connector.
- Work with the Simple Mail Transfer Protocol (SMTP) target connector.
- Work with the Post Office Protocol 3 (POP3) 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:



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 each of the delivered listening connectors, 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 6, "Using Listening Connectors and Target Connectors," Working With the PeopleSoft Connectors, page 85.
HTTP listening connector	This connector provides a web-standard method of communicating with the gateway. It accepts HTTP requests using the GET and POST methods. It also accepts secure HTTPS requests if SSL encryption is configured on the gateway's web server. See Chapter 6, "Using Listening Connectors and Target Connectors," Working With the HTTP Connectors, page 87.

Connector	Description
PeopleSoft 8.1 listening connector	This connector enables PeopleSoft 8.1x applications to communicate with the gateway using native Application Messaging technology. Third-party applications can send properly formatted 8.1x application messages to this connector. It also accepts secure HTTPS requests if SSL encryption is configured on the gateway's web server. See Chapter 6, "Using Listening Connectors and Target Connectors," Working With the PeopleSoft 8.1 Connectors, page 101.
JMS listening connector	This connector enables JMS provider systems to communicate with the gateway using standard JMS protocols. See Chapter 6, "Using Listening Connectors and Target Connectors," Working With the JMS Connectors, page 102.

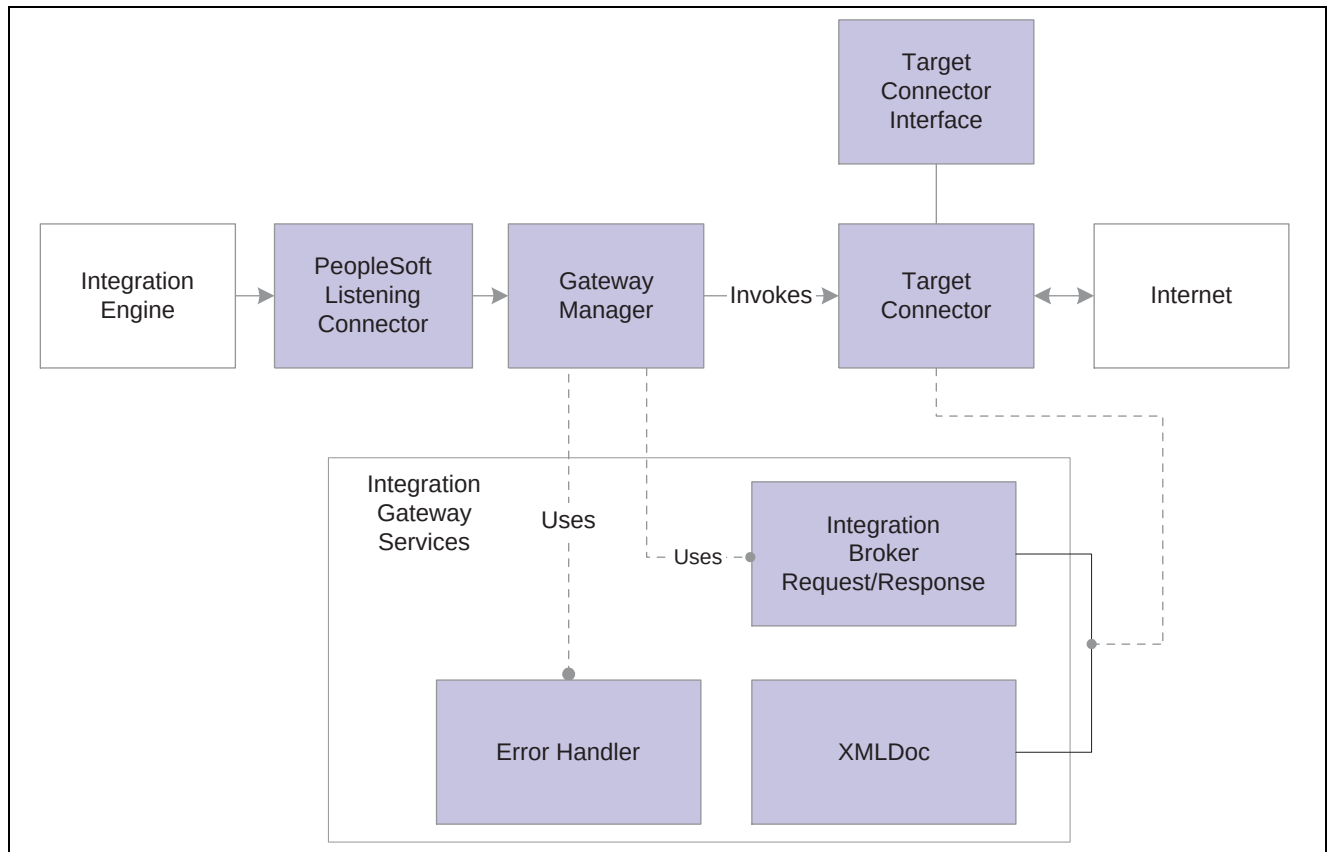
All of the delivered listening connectors that service HTTP requests run as servlets and are configured to run in BEA WebLogic or IBM WebSphere web server environments. These connectors are the PeopleSoft listening connector, the HTTP listening connector, and the PeopleSoft 8.1 listening connector.

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:



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:

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 6, "Using Listening Connectors and Target Connectors," Working With the PeopleSoft Connectors, page 85.
HTTP target connector	This connector provides a web-standard method for the gateway to communicate with PeopleSoft and third-party applications. It sends HTTP requests using the GET and POST methods. It also sends secure HTTPS requests if SSL encryption is configured on the gateway. See Chapter 6, "Using Listening Connectors and Target Connectors," Working With the HTTP Connectors, page 87.
PeopleSoft 8.1 target connector	This connector enables the gateway to communicate with PeopleSoft 8.1x applications that use Application Messaging technology. It converts outbound messages to the Application Messaging native format. It also sends secure HTTPS requests if SSL encryption is configured on the gateway. See Chapter 6, "Using Listening Connectors and Target Connectors," Working With the PeopleSoft 8.1 Connectors, page 101.
JMS target connector	This connector enables the gateway to communicate with JMS provider systems using standard JMS protocols. See Chapter 6, "Using Listening Connectors and Target Connectors," Working With the JMS Connectors, page 102.
FTP target connector	This connector enables the gateway to transfer messages to an FTP server. It converts outbound messages to file data it can send using the FTP PUT command. You can also send messages over a secure FTP(S) protocol. In addition you can receive messages from FTP servers using the GET command. See Chapter 6, "Using Listening Connectors and Target Connectors," Working With the FTP Target Connector, page 116.
SMTP target connector	With this connector, the gateway can send messages to an SMTP server using the PUT command, or receive messages using the GET command. See Chapter 6, "Using Listening Connectors and Target Connectors," Working With the SMTP Target Connector, page 122.

Connector	Description
POP3 target connector	With this connector, the gateway retrieves messages from a POP3 server. Note that although the messages are inbound, the transaction is initiated as an outbound request by the integration engine. See Chapter 6, “Using Listening Connectors and Target Connectors,” Working With the POP3 Target Connector, page 123.
Simple file target connector	With this connector, the gateway saves outbound messages as XML files. See Chapter 6, “Using Listening Connectors and Target Connectors,” Working With the Simple File Target Connector, page 115.
GetMail target connector	This connector provides functionality specific to the PeopleSoft Multichannel Framework. See <i>Enterprise PeopleTools 8.45 PeopleBook: PeopleSoft MultiChannel Framework</i>, “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 4, “Managing Integration Gateways,” Administering Integration Gateways, page 36.](#)

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 9, “Configuring Nodes and Transactions,” Configuring Nodes, page 196.](#)

When you define a transaction for a given node, 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 9, “Configuring Nodes and Transactions,” Configuring Transactions, page 204.](#)

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

Passwords

You must encrypt all required and optional target connector passwords.

See [Chapter 4, “Managing Integration Gateways,” Encrypting Passwords, page 44.](#)

Properties for HTTP URLs

The following connectors communicate over HTTP:

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

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.
- 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 9, “Configuring Nodes and Transactions,” page 195.](#)

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

Working With the PeopleSoft Connectors

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

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

Understanding the PeopleSoft Connectors

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

Using the PeopleSoft Listening Connector

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

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

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

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

Using the PeopleSoft Target Connector

The PeopleSoft target connector initiates conversation with a PeopleSoft application's integration engine over a 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 4, "Managing Integration Gateways," Setting General Connection Properties, page 46](#).

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 base64-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
MessageName	Specify the name of the message.
MessageType	(Optional) Specify the type of message that is sent. Values are: • <i>Sync</i>: The message is synchronous. • <i>Async</i>: The message is asynchronous. • <i>Ping</i>: The message is used to ascertain whether the target node is active or inactive.

Parameter	Description
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 4, "Managing Integration Gateways," Setting General Connection Properties, page 46.
SubChannel	(Optional) Specify the name of a partitioning subchannel to be created at runtime for the message. All messages with the same value for this parameter will be processed in the same subchannel. Unlike the subchannels created by selecting partitioning fields in a message channel definition, the subchannel 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 7, "Defining Message Channels and Messages," Applying Channel Partitioning, page 140.

Parameter	Description
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.
MessageVersion	(Optional) Specify which version of the message is sent.
ExternalMessageID	(Optional) Unique identifier for a message. The ID must not exceed 70 characters. See Using External Message IDs later in this section.

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

See [Chapter 6, “Using Listening Connectors and Target Connectors,” Complying With Message Formatting and Transmission Requirements, page 92.](#)

Using External Message IDs

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

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

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

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

```
http://localhost/PSIGW/HttpListeningConnector?From=QE_UNDERDOG&To=
QE_LOCAL&MessageName=QE_SYNC_MSG&MessageVersion=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:

```
SOAPAction: #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>
<MessageName>QE_SYNC_MSG</MessageName>
<MessageType>sync</MessageType>
<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*.

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.

- MessageName
- MessageType
- OrigTimeStamp
- NonRepudiation
- To
- From

Gateway-Level Connector Properties

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

See [Chapter 6, “Using Listening Connectors and Target Connectors,” Running Integration Gateways Behind Proxy Servers, page 100.](#)

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: • <i>POST</i> (the default). • <i>GET</i>.
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: • <i>Y</i>: Send messages decompressed and decoded. (Default.) • <i>N</i>: Send messages compressed and base64 encoded.
HEADER	Proxy-Authenticate	Specify the user ID and password for proxy authentication. See Chapter 6, “Using Listening Connectors and Target Connectors,” Running Integration Gateways Behind Proxy Servers, page 100.

Property ID	Property Name	Description
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 "+".
- 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>]]>
```

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>
  <MessageName>psft_message_name</MessageName>
  <From>
    <RequestingNode>psft_node_name</RequestingNode>
  </From>
  <ContentSections>
    <ContentSection>
      <Data>
        <![CDATA[<?xml version="1.0"?>your_message_content]]>
      </Data>
    </ContentSection>
  </ContentSections>
</IBRequest>
```

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

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

```
<?xml version="1.0"?>
<IBRequest>
  <MessageName/>
  <MessageType/>
  <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):

```

MessageName: MessageName
MessageType: 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! 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?&MessageName=MessageName
&MessageType=&From=RequestingNode&Password=&OrigUser=&OrigNode=&OrigProcess=
&OrigTimeStamp=&FinalDestination=&To=&SubChannel=&NonRepudiation=&MessageVersion=

```

The full URL query string format is:

```

http://gatewayserver/PSIGW/HttpListeningConnector?&MessageName=MessageName
&MessageType=[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.

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>
    <MessageVersion/>
    <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/xmlsig-core/>

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

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

See [Chapter 17, “Setting Up Secure Messaging Environments,” Implementing Authentication and Nonrepudiation, page 394.](#)

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, 09-Dec-2003 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 9, 2003 at 1:46 p.m. Greenwich Mean Time (GMT).

See [Chapter 8, “Sending and Receiving Messages,” Handling Cookies, page 168.](#)

Responses to Inbound Request Messages

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

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

The integration engine generates and passes back through the listening connector a response message in a format determined by the applicable node, transaction, relationship, and transaction modifier definitions for the request message.

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

The following example shows a successful asynchronous acknowledgment:

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

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

The following is an example of this standard error response:

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

Submitting SOAP Messages

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

```
#MessageName#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: #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: #PURCHASE_ORDER#MY_NODE##PSFT_NODE
```

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

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

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

Understanding HTTP Status Codes

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

Status Codes for Non-SOAP Messages

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

- For an asynchronous message, HTTP status codes 200 to 299 indicate a message status of Success.
- For a synchronous message, the HTTP status code 200 indicates a message status of Success.
- HTTP status code 404 indicates that the server has not found anything matching the Request-URI. In this case, an `ExternalSystemContactException` is generated on the integration gateway and the message status goes to Retry.
- HTTP status code 503 indicates that the server is currently unable to handle the request due to temporary server overload or maintenance. In this case, an `ExternalSystemContactException` is generated on the integration gateway and the message status goes to Retry.
- All other HTTP status codes generate an `ExternalApplicationException`. The status of these messages goes to Error.

Status Codes for SOAP Messages

This section summarizes HTTP status codes for SOAP messages.

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

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

- HTTP status code 400 can mean any of the following:
 - InvalidMessageException
 - MessageMarshallingException
 - MessageUnmarshallingException
- HTTP status code 500 can mean any of the following:
 - ExternalSystemContactException
 - ExternalApplicationException
 - GeneralFrameworkException

Running Integration Gateways Behind Proxy Servers

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

Inserted in the HTTP message header of each transaction is a *Proxy-Authenticate* 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
<code>ig.proxyHost</code>	Enter the domain name of the proxy web server; for example: <code>proxy.peoplesoft.com</code>
<code>ig.proxyPort</code>	Enter the port number of the proxy web server; for example: <code>80</code>

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-Authenticate* 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 4, “Managing Integration Gateways,” Using the `integrationGateway.properties` File, page 41](#)

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 17, “Setting Up Secure Messaging Environments,” Setting Up SSL Encryption, page 401](#)

Using the PeopleSoft 8.1 Listening Connector

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

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

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

Using the PeopleSoft 8.1 Target Connector

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

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

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

Gateway-Level Connector Properties

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

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

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

Node-Level Connector Properties

The following table describes the node-level connector properties:

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

JMS Provider Support

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

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.

Conflicting JAR Files in WebSphere

If you use IBM WebSphere as the gateway web server, the current installation of the Pure Internet Architecture installs PeopleSoft-specific JAR files within the WebSphere directory structure. Several of these JAR files are erroneously installed and have no function in the Pure Internet Architecture. If you use the JMS listening and target connectors, these files can conflict with the JMS provider software. To avoid such conflicts, delete the following files from the WebSphere installation:

- jmq.jar
- jms.jar
- jndi.jar

The reside in the following directory:

<PS_HOME>\webserv\peoplesoft\applications\peoplesoft\PSIGW\WEB-INF\lib

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
<code>ig.jms.JMSProvider.JNDIFactory.Weblogic</code>	Specify the JNDIFactory class name for a BEA WebLogic JMS provider. The default value is: <code>weblogic.jndi.WLInitialContextFactory</code>
<code>ig.jms.JMSProvider.JNDIFactory.iPlanet</code>	Specify the JNDIFactory class name for an iPlanet JMS provider. The default value is: <code>com.sun.jndi.fscontext.RefFSContextFactory</code>
<code>ig.jms.JMSProvider.JNDIFactory.MQSeries</code>	Specify the JNDIFactory class for an MQSeries JMS provider. The default value is: <code>com.sun.jndi.fscontext.RefFSContextFactory</code>

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

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

Using the JMS Listening Connector

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

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

Receiving Messages

The JMS listening connector retrieves topics and queues from the `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
<code>ig.jms.Queues</code>	Specify the number of queue listeners to instantiate.
<code>ig.jms.Queue1</code>	Specify the queue name.
<code>ig.jms.Queue1.Provider</code>	Specify the queue provider name.
<code>ig.jms.Queue1.JMSFactory</code>	Specify the JMSFactory name that is bound to JNDI for the queue.
<code>ig.jms.Queue1.MessageSelector</code>	(Optional) Specify the message filter.
<code>ig.jms.Queue1.URL</code>	Specify the JMS provider's URL to JNDI.
<code>ig.jms.Queue1.User</code>	(Optional) Specify the JMS queue user name.
<code>ig.jms.Queue1.Password</code>	(Optional) Specify the JMS queue password. If you choose to specify a password, you must encrypt it. See Chapter 4, "Managing Integration Gateways," Encrypting Passwords, page 44 .
<code>ig.jms.Queue1.MessageName</code>	(Optional) Specify the name of the message.
<code>ig.jms.Queue1.MessageVersion</code>	(Optional) Specify the message version.
<code>ig.jms.Queue1.RequestingNode</code>	(Optional) Specify the name of the requesting node.
<code>ig.jms.Queue1.DestinationNode</code>	(Optional) Specify the name of the destination node.
<code>ig.jms.Queue1.NodePassword</code>	(Optional) Specify the password for the requesting node. If you choose to specify a password, you must encrypt it. See Chapter 4, "Managing Integration Gateways," Encrypting Passwords, page 44 .
<code>ig.jms.Queue1.SubChannel</code>	(Optional) Specify the name of the subchannel. Messages published to this queue go to the subchannel indicated.

JMS Topic Subscriber Properties

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

Property	Description
<code>ig.jms.Topics</code>	Specify the number of topic subscribers to instantiate.
<code>ig. jms.Topic1</code>	Specify the topic name.
<code>ig. jms.Topic1.Provider</code>	Specify the topic provider name.
<code>ig. jms.Topic1.JMSFactory</code>	Specify the JMSFactory name that is bound to JNDI for the topic.
<code>ig. jms.Topic1.MessageSelector</code>	(Optional) Specify the message filter.
<code>ig. jms.Topic1.URL</code>	Specify the JMS provider's URL to JNDI.
<code>ig. jms.Topic1.User</code>	(Optional) Specify the JMS topic user name.
<code>ig. jms.Topic1.Password</code>	(Optional) Specify the JMS topic password. If you choose to specify a password, you must encrypt it. See Chapter 4, "Managing Integration Gateways," Encrypting Passwords, page 44 .
<code>ig.jms.Topic1.MessageVersion</code>	(Optional) Specify the message version.
<code>ig.jms.Topic1.RequestingNode</code>	(Optional) Specify the name of the requesting node.
<code>ig.jms.Topic1.DestinationNode</code>	(Optional) Specify the name of the destination node.
<code>ig.jms.Topic1.NodePassword</code>	(Optional) Specify the password for the requesting node. If you choose to specify a password, you must encrypt it. See Chapter 4, "Managing Integration Gateways," Encrypting Passwords, page 44 .
<code>ig.jms.Topic1.SubChannel</code>	(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
<code>ig.jms.ErrorQueue</code>	Specify the name of queue to which error messages are published.
<code>ig.jms.ErrorQueue-Provider</code>	Specify the name of the JMS provider.
<code>ig.jms.ErrorQueue-User</code>	(Optional) Specify the JMS error queue user name.
<code>ig.jms.ErrorQueue-Password</code>	(Optional) Specify the JMS error queue password. If you choose to specify a password, you must encrypt it. See Chapter 4, “Managing Integration Gateways,” Encrypting Passwords, page 44 .
<code>ig.jms.ErrorQueue-JMSFactory</code>	Specify the queue connection factory name.
<code>ig.jms.ErrorQueue-Url</code>	Specify the JMS provider’s URL to JNDI.

Error Topic Properties

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

Property	Description
<code>ig.jms.ErrorTopic</code>	Specify the name of topic to which error messages are published.
<code>ig.jms.ErrorTopic-Provider</code>	Specify the name of the JMS provider.
<code>ig.jms.ErrorTopic-User</code>	(Optional) Specify the JMS error topic user name.
<code>ig.jms.ErrorTopic-Password</code>	(Optional) Specify the JMS error topic password. If you choose to specify a password, you must encrypt it. See Chapter 4, “Managing Integration Gateways,” Encrypting Passwords, page 44 .

Property	Description
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 message.
RequestingNode	Specify the requesting node name.
FinalDestinationNode	Specify the final destination nodes. If there are no values, set this property to <i>Null</i> .
DestinationNode	Specify the destination node names, separated with commas. If there are no values, set to "" (empty string).
NodePassword	Enter the node password. This password must be encrypted. See Chapter 4, "Managing Integration Gateways," Encrypting Passwords, page 44 .
SubChannel	(Optional) Specify the name of a partitioning subchannel to be created for the message at runtime. All messages with the same value for this parameter are processed in the same subchannel. Unlike the subchannels created by selecting partitioning fields in a message channel definition, the subchannel that you specify here has no qualifying criteria except the name that you enter. Field-based partitioning is not used for messages with this parameter. See Chapter 7, "Defining Message Channels and Messages," Applying Channel Partitioning, page 140 .

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

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

```

<IBRequest>
  <MessageName>JMS_MessageName</MessageName>
  <MessageType>Async_or_Synch</MessageType>
  <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>
      <MessageVersion></MessageVersion>
      <Data></Data>
    </ContentSection>
  </ContentSections>
</IBRequest>

```

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

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

Starting the JMS Listening Connector

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

To start the JMS listening connector using the servlet:

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

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

To start the JMS listening connector manually:

1. Open a browser and enter the following URL:

<http://localhost/PSIGW/JMSListeningConnectorAdministrator?Activity=Start>

2. Press ENTER.

Shutting Down the JMS Listening Connector

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

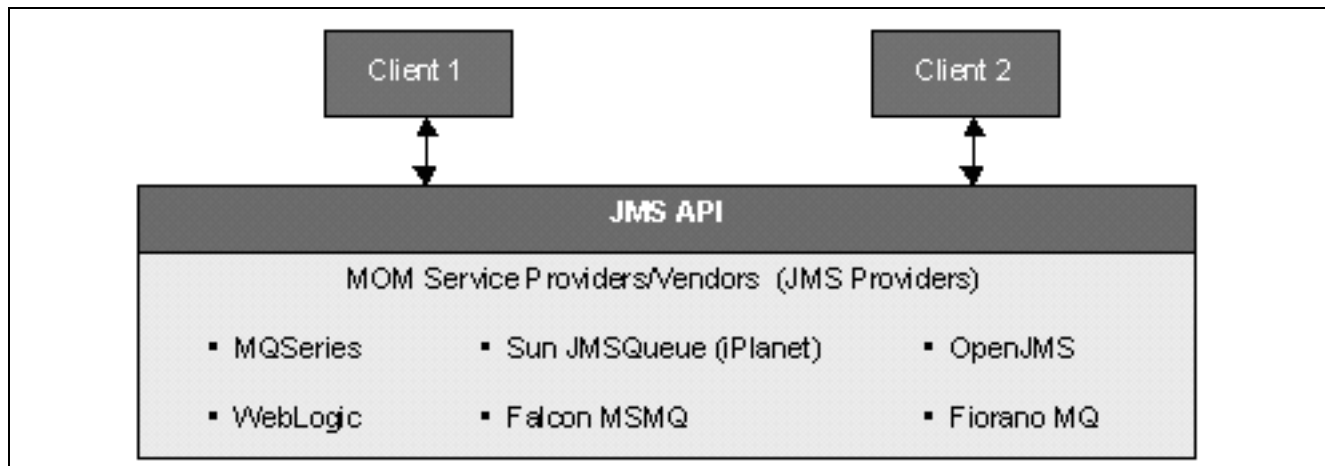
To shut down the JMS listening connector:

1. Open a browser and enter the following URL:
<http://localhost/PSIGW?JMSListeningConnectorAdministrator?Activity=Stop>
2. Press ENTER.

Note. You must register the JMSListeningConnectorAdministrator servlet object under the web application PSIGW in the web server. The BEA WebLogic or IBM WebSphere 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 IBM MQSeries, BEA WebLogic, Sun iPlanet and others. The following diagram illustrates how message flow through the JMS API:



Message flow through the JMS API

The primary features of JMS are.

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

These are defined by separate interfaces so that a provider does not have to support both.

- Topics for publish and subscribe, as well as queues for point-to-point messaging.

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

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

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

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

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

Asynchronous and Synchronous Communication

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

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

For synchronous communication, the exchanges involve only the publisher and a single subscriber. When the response is received, it checks for the JMS correlation ID. If it matches the correlation ID of the message sent, then the JMS target connector processes the response. Consequently, when an external system receives messages, it must set the correlation ID along with its response. The message is not accepted if these two IDs don't match. The target connector verifies that the correlation ID of the response message matches the message ID of the request message. A JMS-compliant remote node receiving a synchronous request message from PeopleSoft must use the value of the message ID of the request message to populate the correlation ID of its response message.

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:

Property ID	Property Name	Description
HEADER	SendUncompressed	Specify whether to send messages decompressed. Values are: • <i>Y</i>: Send the message decompressed and unencoded. This is the default value. • <i>N</i>: Send the message compressed and base 64 encoded.
JMSTARGET	JMSAcknowledgement	Specify the acknowledgment type. Values are: • <i>Auto_Acknowledge</i>. This is the default value. • <i>Client_Acknowledge</i>.
JMSTARGET	JMSDeliveryMode	Specify either durable or nondurable delivery. Values are: • <i>Persistent</i>. • <i>Non-persistent</i> (default).
JMSTARGET	JMSFactory	Specify the factory name. The default value is <i>QueueConnectionFactory</i> .
JMSTARGET	JMSMessageTimeToLive	Specify the time in seconds.

Property ID	Property Name	Description
JMSTARGET	JMSMessageType	Specify the type of message to send. Values are: • <i>Text</i> (default). • <i>MapMessage</i>. • <i>Stream</i>. • <i>Object</i>.
JMSTARGET	JMSMessageTypeClass	(Optional) Specify the implementation class of <i>ProcessJMSMessage</i> . You must set this property when the <i>JMSMessageType</i> is anything other than <i>Text</i> .
JMSTARGET	JMSPassword	(Optional) Specify the password to access the connection. If you choose to specify a password, you must encrypt it. See Chapter 4, “Managing Integration Gateways,” Encrypting Passwords, page 44 .
JMSTARGET	JMS Priority	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: • <i>MQSeries</i> (default). • <i>WebLogic</i>. • <i>iPlanet</i>.
JMSTARGET	JMSQueue	(Optional) Specify the queue name, if you use a queue. You must use and specify either a topic or a queue.

Property ID	Property Name	Description
JMSTARGET	JMSReplyTo	Set this property to <i>True</i> to receive a response from the external system. Values are: • <i>True</i>. • <i>False</i> (default).
JMSTARGET	JMSTopic	(Optional) Specify the topic name, if you use a topic. You must use either a topic or a queue.
JMSTARGET	JMSUrl	Specify the URL.
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 <i>60000</i> (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.
ExternalApplicationException	This exception is generated when: • The correlation ID does not match when the ReplyTo property is set to <i>True</i>. • 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.

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 4, “Managing Integration Gateways,” Encrypting Passwords, page 44](#).

Node-Level Connector Properties

The following table describes the node-level connector properties:

Property ID	Property Name	Description
HEADER	SendUncompressed	Specify whether to save messages decompressed. Values are: • <i>Y</i>: Save the message decompressed and unencoded. This is the default value. • <i>N</i>: Save the message compressed and base64 encoded.
PROPERTY	FilePath	Specify the location where the connector saves the output file. The default location is c:\temp.
PROPERTY	FileName	(Optional) Specify the name of the output file. The file's default name has the following format: <i>sourcenodename.msgname.publication_id.xml</i>
PROPERTY	Method	Specify the method used to send messages. The valid values are: • <i>PUT</i> (the default). • <i>GET</i>.
PROPERTY	Password	(Optional) Specify a password for secure processing. For secure processing, you must also set the <code>ig.fileconnector.password</code> in the <code>integrationGateway.properties</code> 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.

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

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

Prerequisites for Using the FTP Target Connector

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

Specifying Required JAR Files

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

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

Setting Node-Level FTP Connector Properties

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

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

Property ID	Property Name	Description
HEADER	SendUncompressed	Specify whether to send messages decompressed. Values are: • Y: Send messages decompressed and decoded. This is the default value. • N: Send messages compressed and base64 encoded.
FTPTARGET	HOSTNAME	Specify the IP address or name of the FTP server for the connection.
FTPTARGET	METHOD	Specify the method to send or receive messages. The valid values are: • PUT (default). Send messages to an FTP server. • GET. Retrieve messages from an FTP server. • GETDIRLIST. Retrieve a directory list of files from an FTP server.

Property ID	Property Name	Description
FTPTARGET	DIRECTORY	(Optional) Specify the remote directory into which the file is placed. 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. • Message name. • Originating timestamp. 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 4, "Managing Integration Gateways," Encrypting Passwords, page 44.
FTPTARGET	TIMEOUT	Specify the time in milliseconds for the connector to wait for the message to transmit. If the timeout period expires without a successful transmission, the transaction fails. The default value is 50000 (50 seconds).
FTPTARGET	TYPE	Specify the type of file you are sending or receiving. The valid options are: • ASCII (default) • BINARY

Setting Node-Level FTPS Connector Properties

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

Property ID	Property Name	Description
FTPTARGET	FTPS	Enables secure communication over FTP. Values are: • Y: Enable FTPS communication. • N: Disable FTPS communication. This is the default value.
FTPTARGET	CLIENTCERT	(Optional) To use client authentication when establishing a connection with the target or receiving system, enable the CLIENTCERT property.
FTPTARGET	PORT	Specify the port used for communication. The default port is 21.

Using Directory Lists

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

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

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

Example

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

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

See [Chapter 6, “Using Listening Connectors and Target Connectors,” Using Directory Lists, page 119.](#)

```

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

&MSG = CreateMessage(Message.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. */
&yo = &MSG.IBInfo.IBConnectorInfo.AddConnectorProperties("Method", "GET",%Property);
&yo = &MSG.IBInfo.IBConnectorInfo.AddConnectorProperties("HOSTNAME", =>
  "ftp.ftpserver.com",%Property);
&yo = &MSG.IBInfo.IBConnectorInfo.AddConnectorProperties("USERNAME", =>
  "sam",%Property);

/* Encrypt the password */
&pscipher = CreateJavaObject("com.peoplesoft.pt.integrationgateway.common.Encrypt=>
Password");
&encPassword= &pscipher.encryptPassword("ftpserverpassword");
&pscipher = Null;

&yo = &MSG.IBInfo.IBConnectorInfo.AddConnectorProperties("PASSWORD", enc=>
Password,%Property,);
&yo = &MSG.IBInfo.IBConnectorInfo.AddConnectorProperties("DIRECTORY", "/incoming=>
/tmp",);

/* Do Connector Request */
&MSG2 = 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(Message.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 */
&yo = &MSG.IBInfo.IBConnectorInfo.AddConnectorProperties("Method", "GET",%Property);
&yo = &MSG.IBInfo.IBConnectorInfo.AddConnectorProperties("FILENAME", &file⇒
Name,%Property);
&yo = &MSG.IBInfo.IBConnectorInfo.AddConnectorProperties("HOSTNAME", ⇒
"ftp.ftpserver.com",%Property);
&yo = &MSG.IBInfo.IBConnectorInfo.AddConnectorProperties("USERNAME", ⇒
"sam",%Property);

/* Encrypt the password */
&pscipher = CreateJavaObject("com.peoplesoft.pt.integrationgateway.common.Encrypt⇒
Password");
&encPassword= &pscipher.encryptPassword("ftpserverpassword");
&pscipher = Null;

&yo = &MSG.IBInfo.IBConnectorInfo.AddConnectorProperties("PASSWORD", enc⇒
Password,%Property,);
&yo = &MSG.IBInfo.IBConnectorInfo.AddConnectorProperties("DIRECTORY", "/incoming⇒
/tmp",);

/* Do Connector Request */
&MSG3 = ConnectorRequest(&MSG);

```

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.
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.

Property ID	Property Name	Description
HEADER	Content-Type	(Optional) Specify the type of text content that makes up the email body. Values are: • <i>Text/plain</i>. • <i>Text/html</i>.
HEADER	SendUncompressed	Specify whether to send messages decompressed. Values are: • <i>Y</i>: Send the message decompressed and unencoded. This is the default value. • <i>N</i>: Send the message compressed and base 64 encoded.

Working With the POP3 Target Connector

This section provides an overview of the POP3 target connector and discusses how to:

- Work with the default POP3 node definition.
- Pass connector properties as defined.
- Override the defined connector properties.

Understanding the POP3 Target Connector

Use the POP3 target connector to retrieve messages from a POP3 email server. You can use it in two different cases. You can use it when a single user has an account, and you can use it when an application has an account.

Note. The POP3 target connector is a deprecated technology. Its functionality is provided by the GetMail target connector. Existing integrations that use the POP3 target connector will still function without modification. However, PeopleSoft recommends that you develop new integrations to receive email using the GetMail target connector. See the *PeopleTools 8.45 PeopleBook: Multichannel Framework*, “Configuring the Email Channel,” for more information on the GetMail target connector.

The POP3 target connector supports attachments. Before using the attachments feature, you should make sure that the web server and application server have sufficient memory to handle large attachments.

PeopleSoft provides a default node for use with the POP3 target connector, called PT_EMAIL_POP3. More information about this node, including messages and transactions defined for it, is provided later in this section.

The connector ID for the POP3 target connector is *POP3TARGET*.

POP3 Services

When you use the POP3 target connector, you can execute the following predefined services:

Service	Description
ReadAll	Gets all available messages, including information from the following mail fields and mail message sections: From, Reply To, To address collection, Cc address collection, Subject, mail body, and Unique ID Listing (UIDL).
ReadAndDelete	Gets all available messages and deletes them from the server, including information from the following mail fields and mail message sections: From, Reply To, To address collection, Cc address collection, Subject, mail body, and UIDL.
MessageCount	Returns the number of messages in the POP3 server.
ReadMessage	Returns a message for a given UIDL. The response includes information from the following mail fields and mail sections: From, Reply To, To address collection, Cc address collection, Subject, mail body, and UIDL.
DeleteMessage	Deletes a message for a given UIDL.
ReadHeaders	Returns information contained in all available message headers, including information from the following mail fields and mail sections: From, Subject, UIDL, and received date for the mail message. The string <i>NOUIDL</i> is returned for the UIDL part of the message if the POP3 server does not support UIDL. The received date is the date on which the email was sent; it may not be available on all POP3 servers.

Service	Description
ReadHeadersWithAttach	Returns information contained in all available message headers, including information from the following mail fields and mail sections: From, Subject, UIDL, the received date for the mail message, and the message attachment list. The string <i>NOUIDL</i> is returned for the UIDL part of the message if the POP3 server does not support UIDL. The received date is when the email was sent; it may not be available on all POP3 servers.
ReadMessagesWithAttach	Retrieves a specific message with attachments, including information from the following mail fields and mail sections: From, Reply To, To address collection, Cc address collection, Subject, mail body, UIDL, and message attachment list. It also returns a child record that contains the following information: • Content type: If the message is a multipart message, this is the content type for this part. • File name: This is the attachment file name. Set to the content ID if the file name is not available. • File data: The file data available in the attachment. • Message body: If a text or HTML mail message, it includes the message body.

Note. When you use the ReadMessage or DeleteMessage services, you must specify a UIDL value. UIDL is a system used by POP3 servers to uniquely identify a mail message. The UIDL is a fixed string (CHAR, 70) that is unique to the message. The UIDL of a message never changes and is never reused, even when the message has been deleted from the user's mailbox.

Node-Level Connector Properties

The POP3 target connector has the following node-level properties:

Property ID	Property Name	Description
POP3TARGET	Server	Specify the POP3 server name.
POP3TARGET	User	Specify the mailbox user name to use for the POP3 connection.
POP3TARGET	Password	Specify the password for the mailbox user.

Property ID	Property Name	Description
POP3TARGET	Port	Specify the port that the POP3 server uses for the connection. You must specify this value if you're using the user-account approach to pass connector properties through the EMAIL_RES_WRK record. The default is value is <i>110</i>.
POP3TARGET	MethodName	Specify the POP3 service to execute. Values are: • <i>DeleteMessage</i>. • <i>MessageCount</i> (default). • <i>ReadAll</i>. • <i>ReadAndDelete</i>. • <i>ReadHeaders</i>. • <i>ReadHeadersWithAttach</i>. • <i>ReadMessage</i>. • <i>ReadMessageWithAttach</i>. Note. MethodName values are not case-sensitive.
POP3TARGET	Count	Specify the number of messages or headers to fetch. The value <i>0</i> indicates to get all messages or headers. The default is <i>0</i>. This property applies to the following methods: • <i>ReadAll</i>. • <i>ReadHeaders</i>. • <i>ReadHeaderswithAttach</i>. • <i>ReadAndDelete</i>.

The POP3 target connector returns an empty message in addition to the actual message responses for the following MethodName values: *ReadAll*, *ReadDeleteAll*, *ReadHeaders*, *ReadHeadersWithAttach*, and *ReadMessage* services.

You must create separate transactions for the POP3 node for each service; for example, *ReadAll*, *ReadAndDelete*, *MessageCount*, and so forth. For single-user accounts, you set values for server, port, account, user, password, and method name in a request message that contains the EMAIL_RES_WRK work record.

Working With the Default POP3 Node Definition

This section discusses how to:

- Configure the POP3 default node.
- Use the PT_EMAIL_POP3 node.
- Retrieve specific messages from the POP3 server.
- Pass node-level connector properties.

Configuring the POP3 Default Node

The POP3 target connector default node, PT_EMAIL_POP3, has a transaction predefined for each method name value (or service). To configure the node, you must edit each transaction and specify the POP3 server name, user name, and password.

You use the Node Definitions - Transactions page to configure the default node:

Node Name: PT_EMAIL_POP3					
Node Info Contact / Notes Properties Connectors Transactions Portal Content					
Transactions					
Customize Find View All First 1-8 of 8 Last					
Edit	Transaction Type	Request Message	Request Message Version	Effective Date	Status
1 Edit	OutSync	EMAIL_REQ_READMSGATT	VERSION_1	10/24/2001	Active
2 Edit	OutSync	EMAIL_REQ_DELMSG	VERSION_1	09/19/2001	Active
3 Edit	OutSync	EMAIL_REQ_MSGCOUNT	VERSION_1	09/19/2001	Active
4 Edit	OutSync	EMAIL_REQ_READALL	VERSION_1	09/19/2001	Active
5 Edit	OutSync	EMAIL_REQ_READDELALL	VERSION_1	09/19/2001	Active
6 Edit	OutSync	EMAIL_REQ_READHDR	VERSION_1	09/19/2001	Active
7 Edit	OutSync	EMAIL_REQ_READHDRATT	VERSION_1	09/19/2001	Active
8 Edit	OutSync	EMAIL_REQ_READMSG	VERSION_1	09/19/2001	Active

Node Definitions - Transactions page

To set up transactions for PT_EMAIL_POP3:

1. Select PeopleTools, Integration Broker, Node Definitions.
The Nodes page appears.
2. Search for and open the POP3 default node, PT_EMAIL_POP3.
3. Select the Transactions tab.

The predefined transactions for the node appear.

4. Click the Edit link next to the transaction to set up.

Transaction detail tabs appear.

5. Select the Connectors tab.

You may have to click View All link to view all properties for a transaction.

6. Specify values for the following properties:

- Password
- Port
- Server

- User

7. Click Save.

To set up other transactions for the node, click the Return to Transaction List link and repeat steps 4 through 7

Using the PT_EMAIL_POP3 Node

PeopleSoft provides a default node called PT_EMAIL_POP3 for use with the POP3 target connector.

PeopleSoft provides the several message definitions with the PT_EMAIL_POP3 node. The request message name identifies a transaction for the node. The response message name indicates the response returned by the POP3 target connector. For example, when you send the message EMAIL_REQ_MSGCOUNT, it internally executes the service MessageCount and returns the response in the EMAIL_RES_MSGCOUNT message.

Note. The values each message returns depends on the service. The POP3 services are described earlier in this section.

PeopleSoft provides the following message definitions with the PT_EMAIL_POP3 node:

Requesting Message Name	Response Message Name
EMAIL_REQ_MSGCOUNT	EMAIL_RES_MSGCOUNT
EMAIL_REQ_READALL	EMAIL_RES_READALL
EMAIL_REQ_READDELALL	EMAIL_RES_READDELALL
EMAIL_REQ_READMSG	EMAIL_RES_READMSG
EMAIL_REQ_DELMSG	EMAIL_RES_DELMSG
EMAIL_REQ_READHDR	EMAIL_RES_READHDR
EMAIL_REQ_READHDRATT	EMAIL_RES_READHDRATT
EMAIL_REQ_READMSGATT	EMAIL_RES_READMSGATT

All predefined request messages contain the EMAIL_REQ_WRK record. When a request message passes through the POP3 target connector, it retrieves data from the following fields in the EMAIL_REQ_WRK record:

Field	Data Type
User	CHAR (254)
Password	CHAR (254)

Field	Data Type
Server	CHAR (254)
Port	INT
MethodName	CHAR (16)
NUMROWS	INT
UIDL	CHAR (70)

The POP3 target connector then takes the field values from the EMAIL_REQ_WORK and generates a response. The response is contained in the EMAIL_RES_WRK record. The EMAIL_RES_WRK record contains the field values shown in the following table:

Field	Data Type	Description
EMAIL_FROM	CHAR (254)	From.
EMAIL_SENDER	CHAR (254)	Reply to.
UIDL	CHAR (70)	Unique identifier for the email message.
WL_SUBJECT	CHAR (255)	Subject.
EMAIL_TEXTLONG	LONG (512)	Body.
DATE_FROM	DATE	(RECV_DATE).
NUMROWS (Count)	INT	Count of messages used by the EMAIL_REQ_MSGCOUNT and EMAIL_REQ_DELMMSG services.
NOTIFY_TO	LONG Use ' ; ' to separate list	To address collection.

Field	Data Type	Description
NOTIFY_CC	LONG Use ' ; ' to separate list	Cc address collection.
ATTACHMENTS	LONG Use ' ; ' to separate list	Attachment list.

An additional record, EMAIL_ATT_WRK, is returned when there is an attachment. EMAIL_ATT_WRK is a child of EMAIL_RES_WRK. The record EMAIL_ATT_WRK contains the following fields:

Field	Data Type
CONTENT_TYPE	CHAR (254)
FILENAME	CHAR (80)
FILE_DATA	ATTACHMENT
EMAIL_TEXTLONG	LONG

Retrieving Specific Messages from the POP3 Server

You can retrieve specific messages if you are using the ReadMessage, DeleteMessage, and ReadMessageWithAttach services. To retrieve specific messages for these services, set the UIDL in the SyncRequest () method to the value of the message that you want to read. The messages that you specify are located in the EMAIL_RES_MESSAGE table.

To retrieve all messages, enter the value *New* for the UIDL.

Passing Node-Level Connector Properties

Most POP3 requests are performed using the SyncRequest () method, as shown here:

```
SyncRequest ([NodeName])
```

There are two ways to pass node-level properties to the POP3 target connector:

- Pass the values defined for the node as defined in the node definition.
- Override all the values defined in the node definition at runtime.

In this case, you pass property values through method field records.

Passing Connector Properties as Defined

The following code examples demonstrate how to populate the UIDL field before the SyncRequest () call to ensure that PeopleSoft Integration Broker provides a message response in structured message format when no field values are set in the request.

The examples also show the code that you need to add to the end of the request to pass property values via method field records.

MessageCount

The following example shows how to execute SyncRequest for MessageCount:

```
Local Message &MSG;
Local Message &response;

&MSG = CreateMessage(Message.EMAIL_REQ_MSGCOUNT);
&MSG.GetRowset().GetRow(1).GetRecord(Record.EMAIL_REQ_WRK).GetField
(Field.UIDL).Value = "NEW";

&response = &MSG.SyncRequest(Node.PT_EMAIL_POP3);

If (&response.ResponseStatus = 0) Then
    &str = &response.GetRowset().GetRow(1).GetRecord
        (Record.EMAIL_RES_WRK).GetField(Field.NUMROWS).Value
End-If;
```

ReadAll

The following example shows how to execute SyncRequest for ReadAll:

```
Local Message &MSG;
Local Message &response;

&MSG = CreateMessage(Message.EMAIL_REQ_READALL);
&MSG.GetRowset().GetRow(1).GetRecord(Record.EMAIL_REQ_WRK).GetField
(Field.UIDL).Value = "NEW";

&response = &MSG.SyncRequest(Node.PT_EMAIL_POP3);

If (&response.ResponseStatus = 0) Then
    /* get field values here */
End-If;
```

ReadMessage

The following example shows how to execute SyncRequest for ReadMessage:

```
Local Message &MSG;
Local Message &response;

&MSG = CreateMessage(Message.EMAIL_REQ_READMSG);
    &MSG.GetRowset().GetRow(1).GetRecord(Record.EMAIL_REQ_
        WRK).GetField
        (Field.UIDL).Value = &uidl;
    /* UIDL of the message to read */
```

```

&response = &MSG.SyncRequest(Node.PT_EMAIL_POP3);

If (&response.ResponseStatus = 0) Then
/* get field values here */

End-If;

```

DeleteMessage

The following example shows how to execute SyncRequest for DeleteMessage:

```

Local Message &MSG;
Local Message &response;

&MSG = CreateMessage(Message.EMAIL_REQ_DELMSG);
&MSG.GetRowset().GetRow(1).GetRecord(Record.EMAIL_REQ_WRK).
    GetField
(Field.UIDL).Value = &uidl;
/* UIDL of the message to delete */

&response = &MSG.SyncRequest(Node.PT_EMAIL_POP3);

If (&response.ResponseStatus = 0) Then
    /* get status of deleted message in NUMROWS field. */
    /* this is 1 if the message was found and deleted. */

End-If;

```

ReadHeaders

The following example shows how to execute SyncRequest for ReadHeaders:

```

Local Message &MSG;
Local Message &response;

&MSG = CreateMessage(Message.EMAIL_REQ_READHDR);
&MSG.GetRowset().GetRow(1).GetRecord(Record.EMAIL_REQ_WRK).
    GetField
(Field.UIDL).Value = "NEW";

&response = &MSG.SyncRequest(Node.PT_EMAIL_POP3);

If (&response.ResponseStatus = 0) Then
    /* get field values here */

End-If;

```

ReadHeadersWithAttach

The following example shows how to execute SyncRequest for ReadHeadersWithAttach:

```

Local Message &MSG;
Local Message &response;

```

```

&MSG = CreateMessage(Message.EMAIL_REQ_READHDRATT);
&MSG.GetRowset().GetRow(1).GetRecord(Record.EMAIL_REQ_WRK).GetField
(Field.UIDL).Value = "NEW";

&response = &MSG.SyncRequest(Node.PT_EMAIL_POP3);

If (&response.ResponseStatus = 0) Then
    /* get field values here */

End-If;

```

ReadMessageWithAttach

The following example shows how to execute SyncRequest for ReadMessageWithAttach:

```

Local Message &MSG;
Local Message &response;

&MSG = CreateMessage(Message.EMAIL_REQ_READMSGATT);

/* read value from the page */
&str = GetRecord(Record.EMAIL_RES_WRK).GetField(Field.UIDL)
    .Value;
&MSG.GetRowset().GetRow(1).GetRecord(Record.EMAIL_REQ_WRK)
    .GetField
(field.UIDL).Value = &str;

&response = &MSG.SyncRequest(Node.PT_EMAIL_POP3);

If (&response.ResponseStatus = 0) Then
    /* get field values here */
    /* level 0 Rowset */
    &rs = &response.GetRowset();

    /* There is always an empty record, so start from 2 */
    For &i = 2 To &rs.ActiveRowCount

        /* Get each row */
        &row = &rs.GetRow(&i);
        /* level 0 record */
        &rec = &row.GetRecord(Record.EMAIL_RES_WRK);

        &rs1 = &row.GetRowset(Scroll.EMAIL_ATT_WRK);
        &count = &rs1.ActiveRowCount;
        For &j = 1 To &count
            &row1 = &rs1.GetRow(&j);
            &rec1 = &row1.GetRecord(Record.EMAIL_ATT_WRK);
            &str = &rec1.GetField(Field.FILENAME).Value;
            &BI_FILE.WriteLine("Message Part: " | &j);

```

```

        /* Do different action based on Content Type */
        &BI_FILE.WriteLine("Content Type:" |
            &rec1.GetField(Field.CONTENT_TYPE).Value);

        &str = &rec1.GetField(Field.FILENAME).Value;
        &BI_FILE.WriteLine("File Name:" | &str);

        /* Found a file name? - do something with */
        /* the file data*/
        If All(&str) Then

            /* if the file name was not available, */
            /* Content-Id in the message is used, */
            /* this has "<" and ">" in it.*/

            &str = Substitute(&str, "<", "");
            &str = Substitute(&str, ">", "");
            &FILE = GetFile("C:\TEMP\download\FILE" |
                &str,"w", "a", %FilePath_Absolute);

            &FILE.WriteRaw(&rec1.FILE_DATA.Value);
            &FILE.Close();
        Else
            &BI_FILE.WriteLine("File Data: " |
                &rec1.GetField(Field.EMAIL_TEXTLONG)
                .Value);
        End-If;
    End-For;

    &BI_FILE.WriteLine("Message " | &i);

    &str = &rec.GetField(Field.EMAIL_FROM).Value;
    &BI_FILE.WriteLine("From : " | &str);

    &str = &rec.GetField(Field.DATE_FROM).Value;
    &BI_FILE.WriteLine("Recv Date: " | &str);

    &str = &rec.GetField(Field.UIDL).Value;
    &BI_FILE.WriteLine("UIDL: " | &str);

    &str = &rec.GetField(Field.WL_SUBJECT).Value;
    &BI_FILE.WriteLine("Subject : " | &str);
    &str = &rec.GetField(Field.EMAIL_TEXTLONG).Value;
    &BI_FILE.WriteLine("Body: " | &str);
    &str = &rec.GetField(Field.ATTACHMENTS).Value;
    &BI_FILE.WriteLine("Attachments list : " | &str);

    End-For;
Else;
    &BI_FILE.WriteLine("Bad Response");

```



```
End-If;
```

Overriding the Defined Connector Properties

You can override all values defined for the POP3 target connector in the node definition at runtime by passing property values via method field records. The code for each service is the same as described in the previous section. However, you must also pass the following information: POP3 server name, user, password and method name.

To pass property values via method field records, you can use the code examples in the previous section and add the following code to the end of each example:

```
&MSG.GetRowset().GetRow(1).GetRecord(Record.EMAIL_REQ_WRK)
  .GetField(Field.SERVER).Value = <server name of POP3
  server to connect>;
```

```
&MSG.GetRowset().GetRow(1).GetRecord(Record.EMAIL_REQ_WRK)
  .GetField(Field.USER).Value = <user name for the mail a/c
  to get emails>;
```

```
&MSG.GetRowset().GetRow(1).GetRecord(Record.EMAIL_REQ_WRK)
  .GetField(Field.PASSWORD).Value = <password for the user
  used>;
```

```
&MSG.GetRowset().GetRow(1).GetRecord(Record.EMAIL_REQ_WRK)
  .GetField(Field.MethodName).Value = <Service name to be
  executed>;
```

```
/* must specify the port number */
```

```
&MSG.GetRowset().GetRow(1).GetRecord(Record.EMAIL_REQ_WRK)
  .GetField(Field.Port).Value= 110;
```


CHAPTER 7

Defining Message Channels and Messages

This chapter discusses how to:

- Define a message channel.
- Configure a message channel.
- Apply channel partitioning.
- Define a message.
- Define message versions.
- Use records in a message definition.
- Configure message properties.

Understanding Messages

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

- Rowset-based messages.

For hierarchical data 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 Message classes to generate, send, receive, and process these messages.

- Non-rowset-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 Message classes to generate, send, receive and process these messages. If you're handling Simple Object Access Protocol (SOAP) compliant data, you can also use the SoapDoc class to generate and process these messages.

For asynchronous integrations, define a single message. For synchronous integrations, define two messages: one for the request and one for the response.

See Also

[Chapter 8, "Sending and Receiving Messages," page 153](#)

Understanding Message Channels

Channels are logical groupings of messages. Each message must belong to exactly one message channel. Channels are used to:

- Establish message processing sequence.
- Increase throughput by using unordered processing.
- Maintain messaging security.
- Maintain messages with common configuration properties.
- Define processing attributes for timeout parameters, error thresholds, and delivery type.

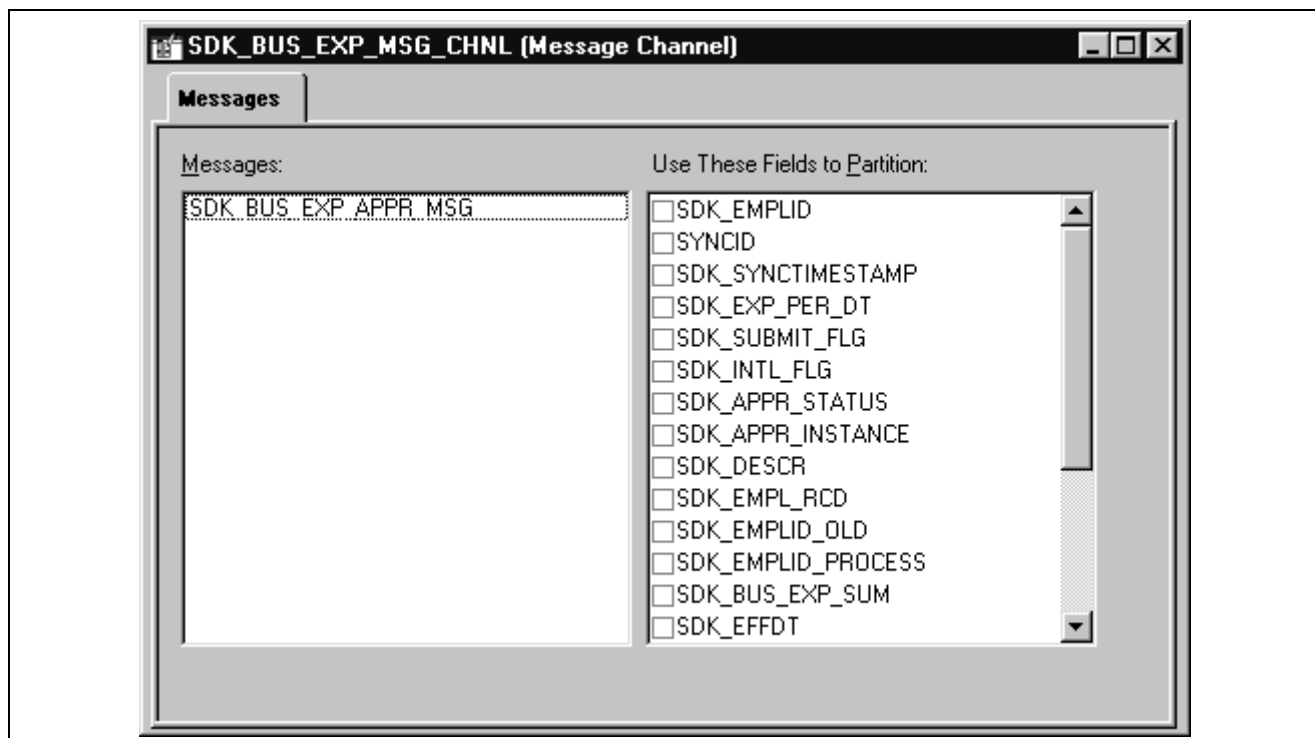
You create channel definitions in PeopleSoft Application Designer. You must define a channel before you define the messages that are assigned to it.

Defining Message Channels

In PeopleSoft Application Designer, select File, New, Message Channel.

A new channel definition appears. It consists of two views, showing assigned messages on the left and partitioning fields on the right. In a new channel definition, both views are blank. You must define messages in PeopleSoft Application Designer before you can assign them to the channel or partition the channel.

The following graphic shows a message channel that contains one message.



Message channel definition

You must save the message channel before you can assign messages to it.

See Also

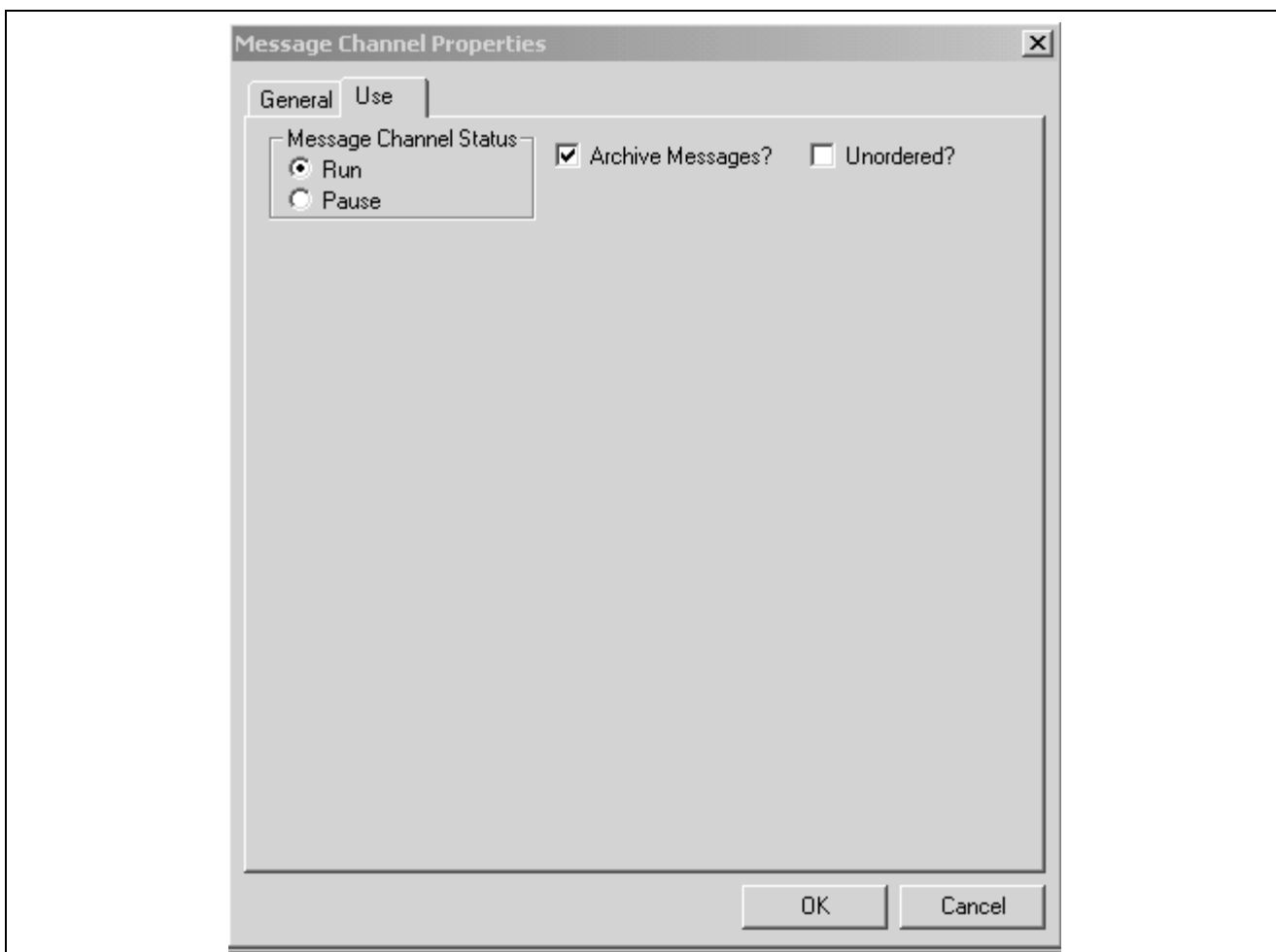
[Chapter 7, “Defining Message Channels and Messages,” Assigning Message Definitions to Message Channels, page 150](#)

[Chapter 7, “Defining Message Channels and Messages,” Applying Channel Partitioning, page 140](#)

[Chapter 7, “Defining Message Channels and Messages,” Granting Security Access to Message Channels, page 142](#)

Configuring Message Channels

To configure a message channel, you use the Use tab in the Message Channel Properties dialog box. To access the Message Channel Properties dialog box, from an open message channel definition, select File, Definition Properties.



Use tab of the Message Channel Properties dialog box.

Message Channel Status

The options are:

Run: Messages assigned to this channel are received and processed normally.

Pause: Messages are received but not processed until the status is reset to *Run*.

Note. Message channels can also be paused and restarted with Integration Broker Monitor.

See [Chapter 14, “Using Integration Broker Monitor,” Monitoring Channel Status, page 311.](#)

Archive Messages?

Select to archive instances of messages that assigned to the channel. 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 Message Details component of the Integration Broker Monitor.

See [Chapter 14, “Using Integration Broker Monitor,” Monitoring Asynchronous Message Details, page 336.](#)

Unordered?

By default, inbound messages assigned to this channel are processed one at a time in the order that 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 when sequence doesn't matter.

Select this check box to force the channel to handle all of its messages 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 channel's messages processed sequentially or if you want to use the partitioning fields.

See [Chapter 7, “Defining Message Channels and Messages,” Applying Channel Partitioning, page 140.](#)

Sending Messages in Order

You can't specify a particular message order as part of the channel definition, even when you clear the Unordered check box and apply partitioning. The channel always processes messages in the order sent. Therefore, be sure to send messages in the proper order.

For example, you can't specify in the channel definition that a message named PO_CREATE is processed before a message named PO_UPDATE. If you send PO_UPDATE before PO_CREATE, the recipient processes PO_UPDATE first.

Applying Channel Partitioning

This section provides an overview of channel partitioning and discusses how to select partitioning fields.

Understanding Channel Partitioning

By default, all inbound messages assigned to a given message channel 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 messages with the same key values. One solution to this inefficiency is partitioning.

Note. Partitioning applies only to asynchronous messaging.

To maximize messaging efficiency and throughput, you want the system to simultaneously process as many messages as possible. Because message channels enforce message sequence, ideally you have a separate message channel for each group of messages that must be processed in order. You can achieve this goal by designating specific fields that are common to the messages assigned to a channel. These fields partition (divide) the channel into subchannels. PeopleSoft Integration Broker creates these subchannels at runtime.

Each subchannel processes only the messages for which the designated common fields have an identical combination of values. The messages within each subchannel are processed in the order that they are sent, so they remain in sequence. Each subchannel works in parallel with the other subchannels—all subchannels simultaneously process their own associated messages.

You implement partitioning by designating the partitioning fields in the message channel definition; no other steps are required.

Note. The more partitioning fields you designate, the more subchannels are generated. If you designate a combination of fields that are unique for each message, all messages are processed simultaneously—in their own subchannels—without regard to sending order. This is the equivalent of selecting the Unordered check box in the message channel properties.

Selecting Partitioning Fields

You can partition a message channel 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 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 messages you assign to the channel, the fewer fields they have in common.

The database record fields common to all messages in the channel appear in the channel's partitioning field list. If only one message is assigned to the channel, 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 common to all messages, regardless of format. If a message channel includes any non-rowset-compatible messages, the message header fields are the only ones that PeopleSoft Integration Broker recognizes as common to every message. You can view them as part of the message XML in Integration Broker Monitor. Some of them are also accessible using equivalent PeopleCode Message class properties, as indicated later in this chapter. The message header fields are:

- *MSGNAME*: This field contains the name of the message definition in PeopleSoft Application Designer.

You can access this field with the Message class *Name* property, which is read-only.

- **MSGNAMEDETAIL:** This field has no automatically defined value—you can assign any value that serves your purposes.

You can access this field with the Message class *MessageDetail* property, which is read-write.

- **PUBLISHER:** This field contains the user ID in effect when the message is published—that is, the ID of the user who is signed on to the publishing database.
- **PUBPROC:** This field refers to the PeopleSoft process that publishes the message.

It's generated when the message 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 channel's partitioning field list. To designate a field as a partitioning key, select the 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 channel's partitioning field list, but they are valid for partitioning.

To designate a message XML field as a partitioning key:

1. Open the message channel definition that you want to partition.
2. Right-click anywhere on the partitioning field list, and select Add Partition Field.
3. Enter the tag name of the XML field and click OK.

The field should exist in all of the channel's assigned messages exactly as named.

Note. When using PublishXmlDoc, defined field tags in messages cannot be mixed case.

At runtime, the integration engine searches each message for the first instance of that field tag and uses the value 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.

Granting Security Access to Message Channels

Users can view messages only in channels to which they have read access. All edit actions require write access.

In addition, in Integration Broker Monitor, users can view only those transactions that contain messages in channels to which they have access.

To grant security access to message channels, use permission lists in the Security component by navigating to PeopleTools, Security, Permissions & Roles, Permission Lists.

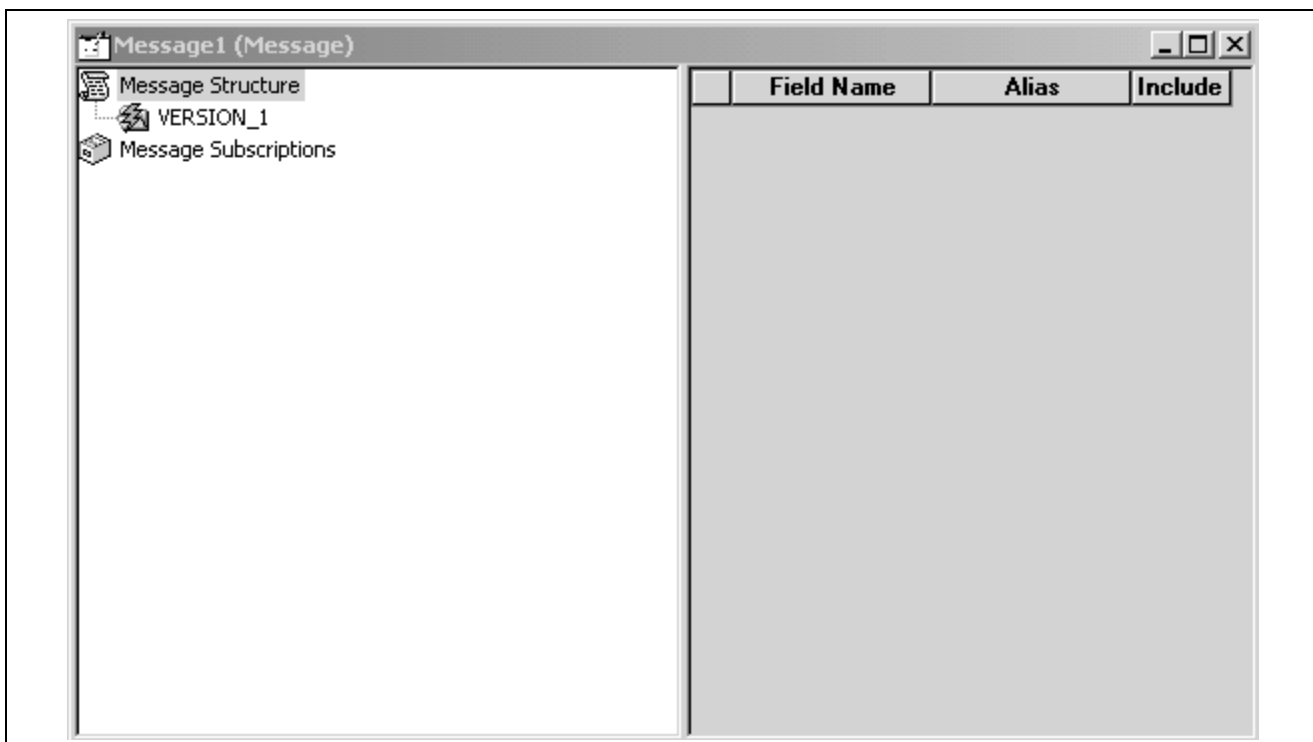
See *Enterprise PeopleTools 8.45 PeopleBook: Security Administration*, “Setting Up Permission Lists”.

Deleting Message Channels During Upgrade

To delete a channel definition in an application upgrade project, first ensure that there are no live instances of messages assigned to that channel. 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.

Defining Message Definitions

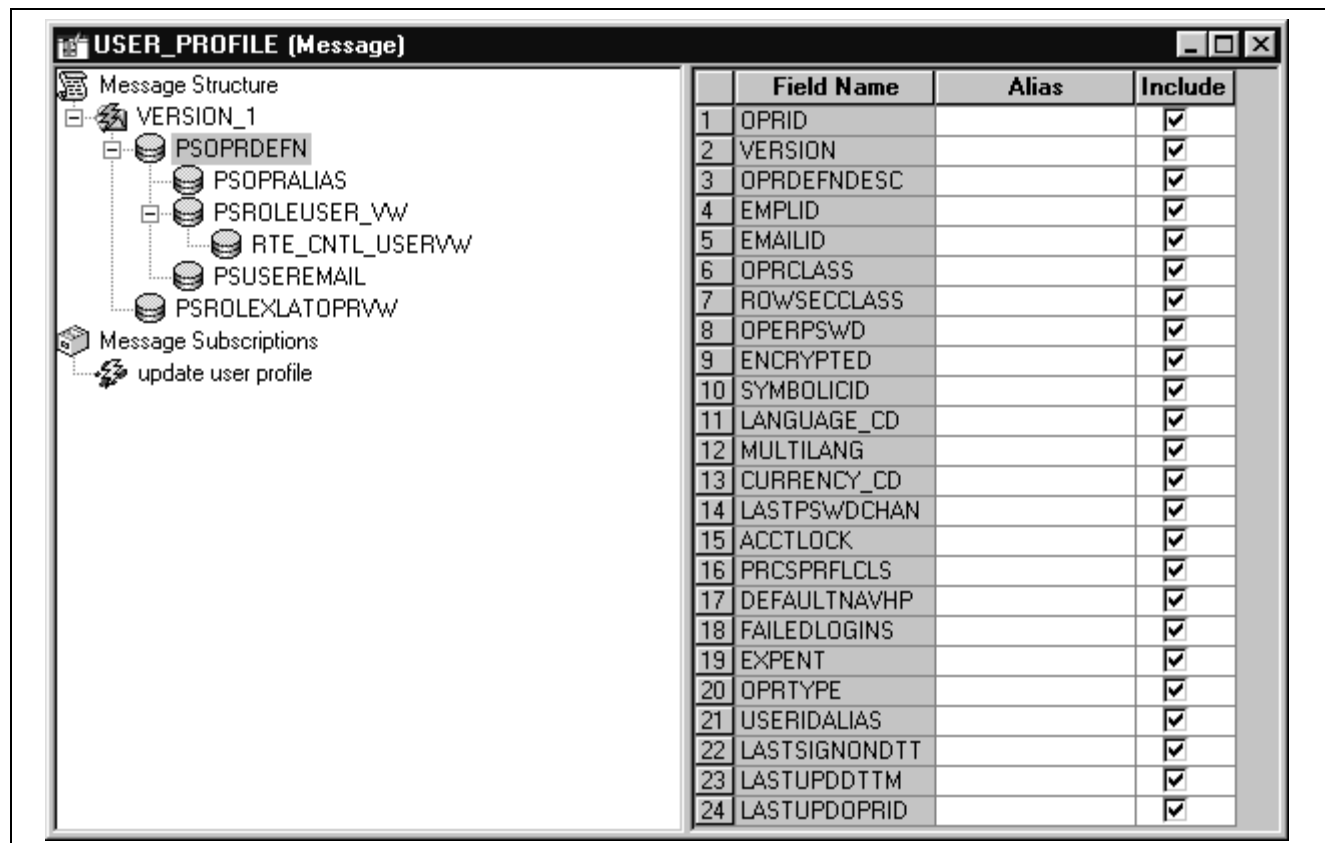
In PeopleSoft Application Designer, select File, New, Message.



New message definition

A new message definition consists of the Message Structure view on the left side of the page and the Field List View on the right side of the page.

The following graphic shows a defined message definition.



User profile message definition

When you click on a record in the Message Structure View, data displays in the Field List View.

To define a new message:

1. Assign the message to a message channel.

Open the message properties dialog box for the new message definition, select the Use tab, specify the message channel that you want to assign it to in the Message Channel field, and click OK.

Note. You can't save the message definition until you assign it to an existing message channel.

2. Save the message definition.

You are prompted for a message name.

Using the Message Structure View

The message structure view includes the following sections:

- Message Structure section.

This section displays different versions of the message and the hierarchy of record definitions within each version. To use the message definition, you must define at least one version; new message definitions supply an initial entry called *VERSION_1* by default. To use the message definition to send or receive PeopleSoft rowset-compatible data, you must populate the version with record definitions.

- Message Subscriptions section.

This section contains the PeopleCode programs used to receive and process inbound asynchronous messages.

Note. To receive and process inbound synchronous messages, PeopleCode must be associated with the message OnRequest PeopleCode event.

Using the Field List View

The field list view displays the fields comprising the selected record definition in the message structure. The field list includes the following columns:

- The Field Name column lists all fields in the selected record definition.
- The Alias column enables you to specify, for each field, an XML field tag value that differs from the original field name.

PeopleSoft Integration Broker sends the field with this name instead of the original field name on the sending system. If the sender doesn't specify an alias value, the field is sent with the original field name. The Alias field is useful for cross release and third-party integration.

- The *Include* column enables you to specify which fields to include in the generated XML.

Select the check box to send the field or to receive it in an inbound message.

See Also

Chapter 5, "Understanding Supported Message Structures," PeopleSoft Rowset-Based Message Format, page 66

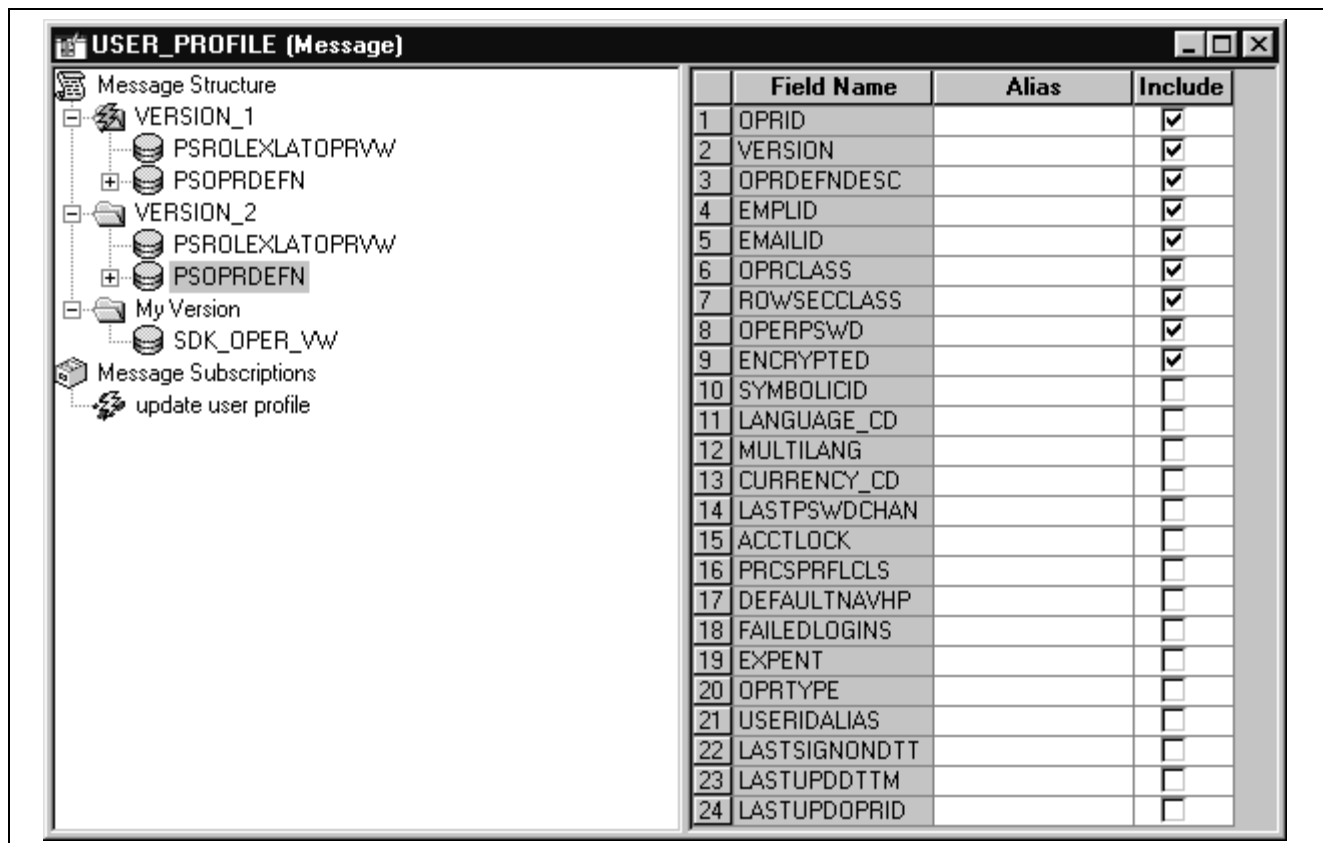
Defining Message Definition Versions

This section discusses how to:

- Create a new message version.
- Rename a message version.
- Set the default version.

Creating New Message Versions

Open an existing message definition.



Message definition containing three versions

With the message definition open, select Insert, Version.

You can also right-click anywhere in the message definition and select Insert Version from the pop-up menu. A new version entry appears on the message structure tree. Note that a corresponding subscription is not added to the Message Subscriptions section. The Subscription or OnRequest PeopleCode must explicitly select the message version that you want to process.

See Also

[Chapter 8, "Sending and Receiving Messages," Understanding Integration PeopleCode, page 155](#)

Renaming Message Versions

Each message version has the default name VERSION_N. This naming scheme starts with VERSION_1 and increases in increments of one for each version that you insert.

To rename a message version, click the message name twice (do not double-click it) and type over the existing name. You can also right-click on the message version name and select Rename from the pop-up menu.

Setting Default Message Versions

The message version marked with the lightning bolt is the default version. To change the default, right-click the version that you want as the default, and select Set As Default from the pop-up menu.

You can also set the default version in the Message Properties dialog box. Select the Use tab, then select the default version from the list of available versions.

See Also

[Chapter 7, “Defining Message Channels and Messages,” Configuring Message Definition Properties, page 148](#)

Using Record Message Definitions

This section provides an overview of message record structure and discusses how to:

- Insert a record into a message definition.
- Reorganize the record structure.
- Create record aliases.

Understanding Message Record Structure

If a message handles PeopleSoft record data, 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 8, “Sending and Receiving Messages,” page 153](#).

You can insert a new record in the message when a version name or a record name is selected in the Message Structure view. If a version name is selected, you can only insert a record immediately below it in the hierarchy—a *child record*. If a record name is selected, you can also insert a *sibling record*—one at the same level as the selected record.

Note. Virtually no difference exists between a child record and a sibling record. You can move any record to any location in the version hierarchy after it has been inserted.

Underlying Record Definitions

Records that you insert in a message definition are proxies for the original record definitions. Any change 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 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.

Inserting Records into Message Definitions

You must insert the first record into a message version as a child record.

To insert a record into a message version:

1. Select a message version or record.

For a child record, select the message version or existing record below which you want to insert the new record. For a sibling record, select an existing record at the same level where you want to insert the new record.

2. Select Insert, Child Record.

For a sibling record, select Insert, Record.

You can also right-click the selected message version or record name and select Insert Child Record (or Insert Record) from the pop-up menu.

3. Use the Insert Record dialog box to search for, select, and insert the record.

Note. The Insert Record dialog box remains active until you click Close. This enables you to quickly insert multiple records at the same level.

Reorganizing Record Structure

Use the directional buttons on the message definition toolbar to reposition a record within the hierarchy of a message version.

The up and down arrows do not change the level of the selected item—they change only its order among other items at that level. The left and right arrows move the selected item lower and higher in the message structure hierarchy.

Note. When you reposition a record in the message version structure, its child records are also repositioned, as well as their child records, and so on.

Creating Record Aliases

In addition to field aliases, you can also provide an alias for each record in a rowset-based message definition. The alias is an XML record tag value which differs from the original record name. PeopleSoft Integration Broker sends the record with this name instead of the original record name on the sending system. If the sender doesn't specify an alias value, the record is sent with the original record name. Record aliases are useful for cross release and third-party integration.

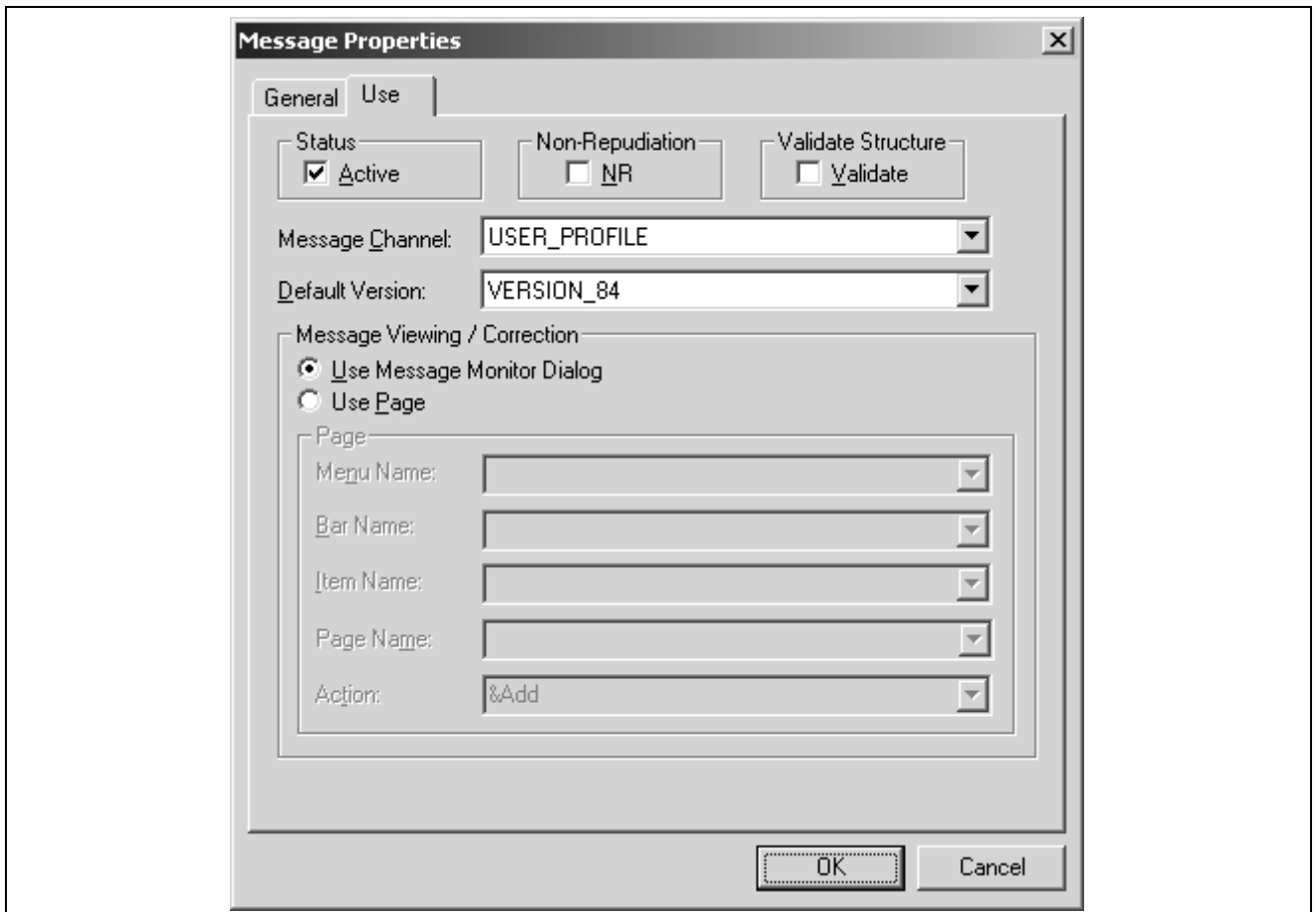
In the message structure view, right-click the record name and select Message Record Properties. In the Message Record Properties dialog box that appears, enter an alias for the record name.

See Also

Chapter 5, “Understanding Supported Message Structures,” PeopleSoft Rowset-Based Message Format, page 66

Configuring Message Definition Properties

With a message definition open, click the Properties button and select the Use tab.



Message Properties dialog box: Use tab

Active

Select to indicate that PeopleSoft Integration Broker can send this message.

Note. If you clear this check box to deactivate the message, any transactions and relationships that use it are also deactivated. Activating the message does not automatically reactivate the transactions and relationships; they must be individually reactivated.

NR (nonrepudiation)

Select to indicate that this message must be sent and received in nonrepudiated form.

Validate

Select to indicate that when the message is received, each field's data type and length should be validated against the field's format in the local database. If a field fails this validation, the transaction fails with an error.

If you clear this check box, validation is still performed, but the transaction doesn't fail. Instead, the data in mismatched fields is truncated to fit the field length in the local definition.

Note. It's recommended that you enable validation to ensure that field formats are identical at the sending and receiving systems. You can nevertheless use different field, record, and message definitions on each system by applying a transform program to the message before sending it or after receiving it.

See [Chapter 12, “Applying Filtering, Transformation and Translation,” page 247.](#)

Message Channel	Select the message channel to which you want to assign the message. To save the message definition, you must first specify a message channel.
Default Version	Select the version of this message that is sent or received by default when the sending or receiving PeopleCode does not specify a version.
Message Viewing/Correction	Specify the message viewing and correction method to use for the message at runtime. The options are: <i>Use Message Monitor Dialog:</i> Use Integration Broker Monitor to view and correct the message. This is the default value. <i>Use Page:</i> Use a page that you design to view and correct the message.

Implementing Nonrepudiation

You must supply the node-based digital certificates containing the private and public keys needed for each transaction. These elements are required at every node that participates in the transaction; PeopleSoft Integration Broker handles the mechanics of applying the keys.

You must configure the nodes that send and receive this message so that they support nonrepudiation.

If a participating node doesn't use PeopleSoft Integration Broker, that node is nevertheless responsible for managing the appropriate public and private keys, inserting properly formatted signatures in the nonrepudiation messages that it sends, and properly handling signatures in messages that it receives.

See Also

[Chapter 17, “Setting Up Secure Messaging Environments,” Implementing Authentication and Nonrepudiation, page 394](#)

Assigning Message Definitions to Message Channels

Before you can assign a message to the message channel, you must define the message. You make the assignment in the message definition's properties.

See [Chapter 7, “Defining Message Channels and Messages,” Defining Message Definitions, page 143.](#)

Note. Once a message is assigned to the channel and appears in the Messages view, you can open its definition by double-clicking the message name.

To assign a message definition to a message channel.

1. In PeopleSoft Application Designer, open a message definition.
2. Click the Properties button and select the Use tab.
3. From the Message Channel drop-down list, select the message channel to which you want to assign the message.
4. Save the changes.

Deleting Message Definitions 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.

CHAPTER 8

Sending and Receiving Messages

This chapter discusses how to:

- Access integration events.
- Create message subscriptions.
- Generate and sending messages.
- Set and override connector properties at runtime.
- Override synchronous timeout intervals.
- Receive and process messages.
- Process inbound errors.
- Use Message object functionality with nonrowset-based messages.
- Generate test messages.

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 channel and message in PeopleSoft Application Designer.

Note. Once you create PeopleCode, you must also define associated nodes and transactions 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.

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 performing.

When a message is sent, it goes 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 or function.
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 transaction definitions that are associated with that message definition to determine the valid target nodes for the message.

The asynchronous process examines only asynchronous transactions, and the synchronous process examines only synchronous transactions. If a process doesn't find a node definition that contains an outbound transaction specifying that message, the sending method returns an error.

Note. Any outbound transaction that is found, the message to which it applies, and the node for which it's defined must all have an active status for the process to proceed.

5. For each outbound transaction 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 or function.

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 the inbound transaction definitions to find one that is associated with the sending node that specifies the same message, version, and transmission type.
3. If a matching transaction isn't found, the integration engine returns an error message through the gateway to the sending node.

If a transaction 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 transaction that is found, the message to which it applies, and the node for which it's defined must all have an active status for the process to proceed.

4. The process accesses the message definition that matches the inbound message name and passes the message to the definition's associated receiving PeopleCode.
 - The asynchronous request process invokes the message definition's subscription PeopleCode.
 - The synchronous request process invokes the message definition's OnRequest 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.
- Routing events.
- Messaging events.
- Messaging PeopleCode.

Sending and Receiving PeopleCode

This section discusses the PeopleCode you use for sending messages from PeopleSoft Integration Broker to other systems, as well as the PeopleCode you use for receiving messages into Integration Broker 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.

In addition to using the Message class for manipulating message content, you also use it to send messages in rowset-based and nonrowset-based formats as described in this section.

PeopleCode used to send messages is highlighted in the following table.

Transmission Type	Message Structure	Sending PeopleCode	Comments
Synchronous	Rowset-based	SyncRequest method.	The SyncRequest method belongs to the Message class.
Synchronous	Nonrowset-based	SyncRequestXMLDoc built-in function.	None.
Asynchronous	Rowset-based	Publish method.	The Publish method belongs to the Message class.
Asynchronous	Nonrowset-based	PublishXMLDoc built-in function.	None.

To work with messages in SOAP format, transform the SOAP documents into XML documents and then use the SyncRequestXMLDoc or PublishXMLDoc built-in functions.

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. Place PeopleCode for synchronous messages in the OnRequest event. Place PeopleCode for asynchronous messages in the Subscription event for the message. You access both of these events in PeopleSoft Application Designer.

Transmission Type	Message Structure	Receiving PeopleCode	Comments
Synchronous	Rowset-based.	GetMessage built-in function.	Enter the PeopleCode in the OnRequest event.
Synchronous	Nonrowset-based	GetMessageXMLDoc built-in function.	Enter the PeopleCode in the OnRequest event.
Asynchronous	Rowset-based	GetMessage built-in function.	Enter the PeopleCode in the Subscription event.
Asynchronous	Nonrowset-based	GetMessageXMLDoc built-in function.	Enter the PeopleCode in the Subscription event.

To get content data out of a message, use the following guidelines.

Message Structure	PeopleCode	Comments
Rowset-based.	GetRowSet	None.
Nonrowset-based	GetXMLDoc	You can also use Message class functionality with nonrowset-based messages. See Chapter 8, “Sending and Receiving Messages,” Using Message Object Functionality With Nonrowset-Based Messages, page 191.

Routing Events

Routing events determine how a message is routed to or from PeopleSoft Integration Broker. Routing events are accessed on message definitions.

OnRouteSend Event

Use OnRouteSend on outbound synchronous and asynchronous messages to specify to what node PeopleSoft Integration Broker routes a message. This event enables you to apply PeopleCode that filters the destination nodes to which PeopleSoft Integration Broker routes messages.

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 message definition's OnRouteSend event, which enables you to apply additional PeopleCode that determines the final target nodes. You can employ this event regardless of how the message is sent.

You can use OnRouteSend to validate the outbound message 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.

See [Chapter 8, “Sending and Receiving Messages,” Accessing Integration Events, page 159.](#)

OnRouteReceive Event

Use OnRouteReceive on inbound synchronous and asynchronous messages to apply PeopleCode that determines whether the default local node accepts the inbound message.

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 message definition's OnRouteReceive event, 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.

See [Chapter 8, “Sending and Receiving Messages,” Accessing Integration Events, page 159.](#)

ReturnToServer Function

Routing events can get and examine messages that are being routed, but they cannot modify message content in any way. Only one statement in the routing code affects the message: the Message class built-in function **ReturnToServer**.

In the routing events, only some of the parameters of ReturnToServer are valid. The following table outlines the effect of each valid parameter:

Routing Event	Statement	Effect
OnRouteSend	ReturnToServer(True);	The outbound message target node list is accepted. The message is sent to all nodes on the list. Note. With more than one target node, an outbound synchronous message fails.
OnRouteSend	ReturnToServer(False);	The outbound message target node list is rejected. The message isn't sent.
OnRouteSend	ReturnToServer(&node_array);	The outbound message target node list is compared to the list in &node_array. The message is sent only to the nodes on both lists.
OnRouteReceive	ReturnToServer(True);	The inbound message is accepted by the default local node.
OnRouteReceive	ReturnToServer(False);	The inbound message is rejected by the default local node.

Outbound Routing PeopleCode Example

The following OnRouteSend example routes the message according to a newly generated target node list that's based on a field value in the message:

```
local message &MSG;
local array &NODE_ARRAY;

&MSG = GetMessage();
&EMPLID = &MSG.GetRowset()(1).QA_INVEST_HDR.EMPLID.Value;

&SELECT_SQL = CreateSQL("select PUBNODE from PS_EMPLID_NODE
    where EMPLID = :1", &EMPLID);

&NODE_ARRAY = CreateArray();
While &SELECT_SQL.Fetch(&PUBNODE)
    &NODE_ARRAY.Push(&PUBNODE);
End-While;

ReturnToServer(&NODE_ARRAY);
```

Inbound Routing PeopleCode Example

The following OnRouteReceive example accepts the message if it contains a field value that's matched in the local database:

```
local message &MSG;

&MSG = GetMessage();
&BUSINESS_UNIT = &MSG.GetRowset()(1).PO_HDR.BUSINESS_UNIT.Value;

SQLExec("Select BUSINESS_UNIT From PS_BUSINESS_UNIT
    where BUSINESS_UNIT = :1", &BUSINESS_UNIT, &FOUND);

If all(&FOUND) Then
    ReturnToServer(True);
Else
    ReturnToServer(False);
End-if;
```

Messaging Events

This section describes events used in messaging.

Outbound Messaging Events

This section describes events used on outbound messages from PeopleSoft to other systems.

OnSend	Use on outbound synchronous and asynchronous messages to override connector properties before sending a message to the integration gateway.

Subscription Use on outbound asynchronous messages when another system subscribes to a message. This event is used for subscription processing, and for validating and loading message data.

See [Chapter 8, “Sending and Receiving Messages,” Creating Message Subscriptions, page 161.](#)

Inbound Messaging Events

This section describes events used on inbound messages to PeopleSoft from other systems.

OnRequest Use on inbound synchronous messages when a response is required.

OnAckReceive Use on inbound asynchronous messages to access the body of a message acknowledgement to check for SOAP faults.

See [Chapter 8, “Sending and Receiving Messages,” Handling Inbound Asynchronous Transactions, page 171.](#)

Messaging PeopleCode

Messaging PeopleCode enables you to manipulate message content. The three messaging PeopleCode classes you can use for this purpose are:

Message class Use for rowset or nonrowset-based messages.

XMLDoc class Use for XML DOM-compliant messages.

SOAPDoc class Use for SOAP-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 class.

See Also

[Chapter 8, “Sending and Receiving Messages,” Using Message Object Functionality With Nonrowset-Based Messages, page 191](#)

Enterprise PeopleTools 8.45 PeopleBook: PeopleCode API Reference, “Message Class”

Enterprise PeopleTools 8.45 PeopleBook: PeopleCode API Reference, “SOAPDoc Class”

Enterprise PeopleTools 8.45 PeopleBook: PeopleCode API Reference, “XmlDoc Class”

Accessing Integration Events

You access integration events in PeopleSoft Application designer, either from the Message Structure section or from the Message Subscription section of the message definition. The following table lists the integration events and in what section of the message definition you access them.

Integration Event	Where Accessed
OnRouteSend	Message Structure section.
OnRouteReceive	Message Structure section.
OnRequest	Message Structure section.
OnAckReceive	Message Structure section.
OnSend	Message Structure section.
Subscription	Message Subscription section.

Accessing Integration Events From the Message Structure Section

To access integration events from the Message Structure section:

1. In PeopleSoft Application Designer, open the message definition for which you want to apply routing.
2. Right-click anywhere in the Message Structure section of the message structure view.
3. Select View PeopleCode.

The PeopleCode editor for the message definition appears. Make sure the message name appears in the left drop-down list.

4. Select the event with which to work from the drop-down list on the right. The valid options are:
 - OnRouteSend
 - OnRouteReceive
 - OnRequest
 - OnAckReceive
 - OnSend

The PeopleCode editor for the selected event appears. Enter or modify the program code.

See Also

Enterprise PeopleTools 8.45 PeopleBook: PeopleCode Developer's Guide, "Using the PeopleCode Editor"

Accessing Integration Events From the Message Subscription Section

To access integration events from the Message Subscription section:

1. In PeopleSoft Application Designer, open the message definition for which you want to apply routing.
2. Right-click the message subscription under the Message Subscriptions section and select View PeopleCode.

The PeopleCode editor for the message subscription appears. To verify that you are accessing the Subscription event, Subscription should appear in the drop-down list on the right side of the editor. Modify the program code as desired.

If no message subscription PeopleCode program name displays under the Message Subscriptions section, no subscription PeopleCode program has been created for the message definition.

See Also

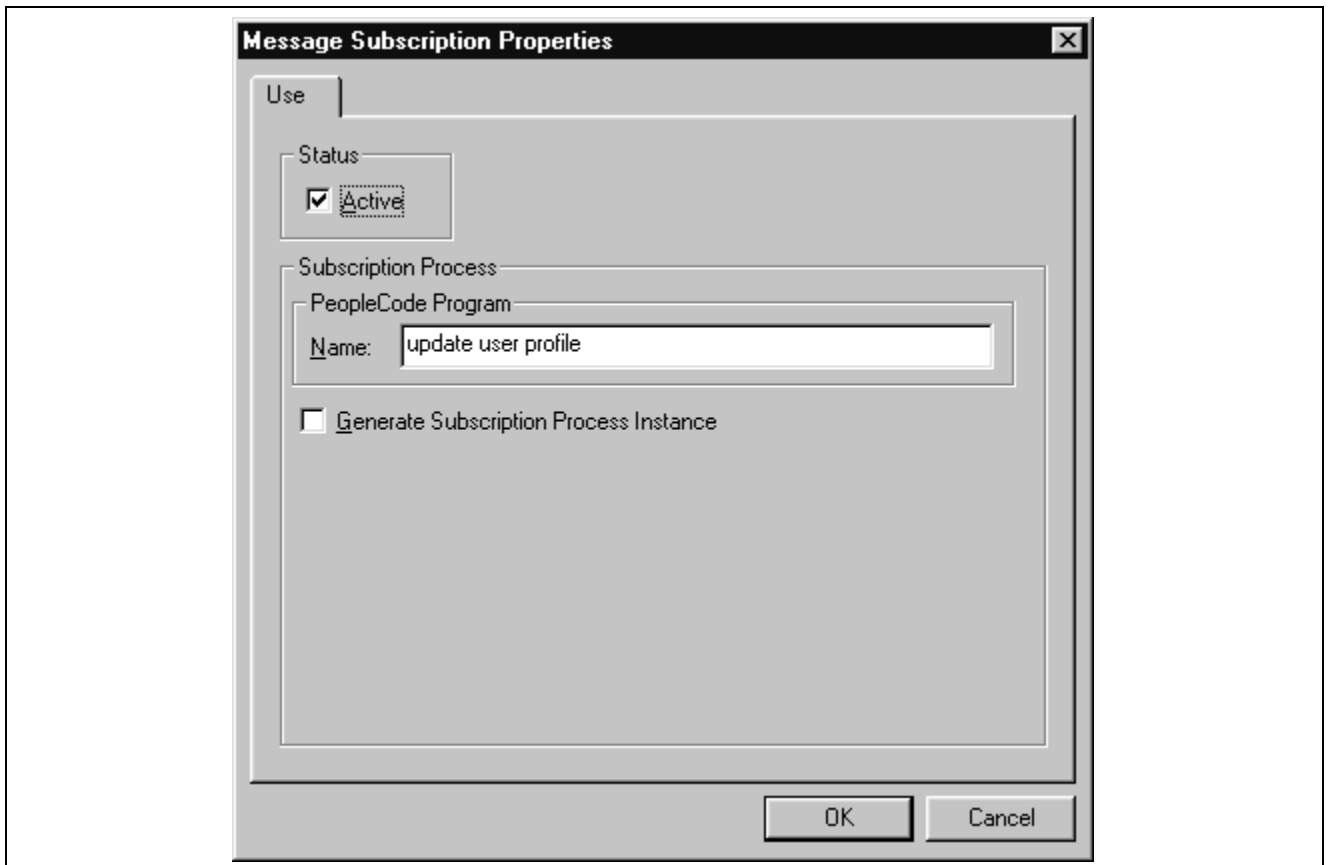
[Chapter 8, “Sending and Receiving Messages,” Creating Message Subscriptions, page 161](#)

Enterprise PeopleTools 8.45 PeopleBook: PeopleCode Developer’s Guide, “Using the PeopleCode Editor”

Creating Message Subscriptions

You can create multiple message subscriptions for a message, each with a different name. Each subscription runs parallel with the others, processing its own copy of the message. In this way, a single inbound message can simultaneously be processed for different purposes.

The only purpose of a message subscription is to represent a PeopleCode program associated with the Subscription event. An event does not fire if you do not define at least one message subscription.



Message Subscription Properties dialog box

To create a message subscription:

1. Open a message definition in PeopleSoft Application Designer.
2. Right-click anywhere in the message structure view and select Insert Message Subscription.

The Message Subscription Properties dialog box appears.

3. Select the Active check box to enable the subscription.
4. Enter a name for the subscription program.
5. Select the Generate Subscription Process Instance check box to automatically produce a unique number that identifies the process request each time that PeopleSoft Process Scheduler runs the subscription in batch mode.
6. Click OK.

The Subscription PeopleCode Editor for the subscription appears, and you can write the asynchronous subscription program.

Note. You can modify the subscription properties at any time. Right-click on the subscription name in the Message Structure view and select Message Subscription Properties to open the Message Subscription Properties dialog box.

See Also

Enterprise PeopleTools 8.45 PeopleBook: PeopleCode Developer's Guide, "Using the PeopleCode Editor"

Generating and Sending Messages

This section provides an overview of outbound messaging and discusses how to handle:

- Outbound asynchronous transactions.
- Outbound synchronous transactions.
- 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 they were sent and that they're single-threaded at the PeopleSoft receiving node. However, message order is not part of the channel definition. You must send messages in the proper order.

Note. You can modify this behavior by using channel partitioning.

See [Chapter 7, "Defining Message Channels and Messages," Applying Channel Partitioning, page 140](#).

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, then viewing it in the Integration Broker Monitor. From the Integration Broker Monitor, you can view the contents of each field to verify that the message data is formatted correctly.

See [Chapter 14, "Using Integration Broker Monitor," page 291](#).

You can also trigger a message with the Test Message feature in PeopleSoft Application Designer. See [Chapter 8, “Sending and Receiving Messages,” Generating Test Messages, page 192.](#)

Message Class Outbound PeopleCode

Use `SelectByKey` 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.

Use `If &Msg.Size > %MaxMessageSize` at Level 0 to control message size when dealing with multiple transactions.

Note. The `OnRouteSend` event that is associated with the message definition enables you to apply PeopleCode that filters the destination nodes.

See [Chapter 8, “Sending and Receiving Messages,” Accessing Integration Events, page 159.](#)

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]]>
```

Handling Outbound Asynchronous Transactions

Use the `Publish` method in the PeopleCode Message class or the `PublishXmlDoc` built-in function. To send a message asynchronously, you can employ `Publish` or `PublishXmlDoc` in:

- A record field PeopleCode event.
- A component PeopleCode event.
- An Application Engine program.
- A message subscription program.
- The message `OnRequest` PeopleCode event.

Note. Message subscriptions and the `OnRequest` event are triggered only when an inbound transaction occurs, but when the receiving PeopleCode runs, it can also send messages.

Message Class Outbound Asynchronous Example

The following example uses the `Publish` method in the PeopleCode Message class:

```
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(Message.SALES_ORDER_ASYNC);
```

```

/*Copy the rows from the rowset to the message object */
&MSG.CopyRowset(&SALES_ORDER);
/*Send the message */
&MSG.Publish();

```

XmlDoc Class Outbound Asynchronous Example

The following example uses the PublishXmlDoc built-in function in the PeopleCode XmlDoc class:

```

Local XmlDoc &xmlRequestDoc;

/* Create an XmlDoc object */
&xmlRequestDoc = CreateXmlDoc();

/* Parse an input XML file into an XmlDoc */
&bool = &xmlRequestDoc.ParseXmlFromURL("C:\pt\appserv\files\input.xml");

/* Send request */
PublishXmlDoc(&xmlRequestDoc, Message.MyMessage, Node.MyNode);

```

Identifying SOAP Faults

You can use the PeopleCode event OnAckReceive 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 &MSG.GetXmlDoc() to get the response data. This returns an XmlDoc object.

If the message is rowset-based, use &MSG.GetXMLString() 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 &MSG.GetXmlDoc() and &MSG.GetXMLString() in the OnAckReceive event. Valid status overrides are %Message_Done, %Message_Error, and %Message_Retry.

```

Local Message &MSG;
Local string &str;
Local XmlDoc &xmlDoc;

&MSG = GetMessage();
/* if message is nonrowset-based */
&xmlDoc = &MSG.GetXmlDoc();

/* if message is rowset-based */
&str = &MSG.GenXMLString();

ReturnToServer(%Message_Done);

```

You can also use the OnAckReceive PeopleCode event to read response content data returned from third-party systems when using the HTTP target connector.

See Also

Enterprise PeopleTools 8.45 PeopleBook: PeopleCode API Reference, “XmlDoc Class”

Handling Outbound Synchronous Transactions

Use the SyncRequest method in the PeopleCode Message class or the SyncRequestXmlDoc built-in function. To send a message synchronously, you can employ SyncRequest or SyncRequestXmlDoc 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.
- The message OnRequest PeopleCode event.
- A message subscription program.

Note. The OnRequest event and subscription programs are triggered only when an inbound transaction occurs, but 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 8, “Sending and Receiving Messages,” Receiving and Processing Messages, page 171.](#)

Message Class Outbound Synchronous Example 1

The following example uses the SyncRequest method in the PeopleCode Message class:

```
Local Message &MSG, &response;
Local Rowset &SALES_ORDER;

&SALES_ORDER = GetLevel0();
&MSG = CreateMessage(Message.SALES_ORDER_SYNC);
&MSG.CopyRowsetDelta(&SALES_ORDER);

/* send the synchronous request; the return value is the response message object */
&response = &MSG.SyncRequest();

/* check the response status; 0 means OK */
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(Message.ZIP_REQUEST);
&msgZipRequest.CopyRowset(&RS);
/* send the synchronous request; the return value is the response message object =>
*/
&msgZipResponse = &msgZipRequest.SyncRequest(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 SyncRequestXmlDoc built-in function:

```
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;

/* Parse the input XML string into an XmlDoc */
&xmlRequestDoc = CreateXmlDoc(&xmlString);

/* Send request */
&xmlResponseDoc = SyncRequestXmlDoc(&xmlRequestDoc, Message.XMLSYNCREQ);

/* Display the output XML string */
&DescrLong.Value = &xmlResponseDoc.GenXmlString();
```

SoapDoc Class Outbound Synchronous Example

The following example uses the SyncRequestXmlDoc built-in function.

Note. Because SyncRequestXmlDoc is an XmlDoc built-in function, you must convert the SoapDoc request message to an XmlDoc object before sending it. Likewise, the response message is an XmlDoc object, so you must convert it to a SoapDoc object to process it with SOAP methods.

```

Local File &MY_FILE;

Local XmlDoc &request, &response;
Local string &strXml;
Local SoapDoc &soapReq, &soapRes;

&soapReq = CreateSoapDoc();
&bool = &soapReq.ParseXmlFromURL("C:\pt\appserv\files\inputSoap.xml");

&strXml = &soapReq.GenXmlString();
&MY_FILE = GetFile("c:\temp\syncInput.txt", "w", "a", %FilePath_Absolute);
&MY_FILE.WriteLine("YEAH!");
&MY_FILE.WriteLine(&strXml);
&MY_FILE.Close();

/* Convert request from SoapDoc to XmlDoc before sending */
&request = &soapReq.XmlDoc;
&response = SyncRequestXmlDoc(&request, Message.QE_SOAP_REQ, Node.UNDERDOG);

&soapRes = CreateSoapDoc();
/* Convert response from XmlDoc back to SoapDoc */
&soapRes.XmlDoc = &response;
&OK = &soapRes.ValidateSoapDoc();
&strXml = &soapRes.GenXmlString();

&MY_FILE = GetFile("c:\temp\sync.txt", "w", "a", %FilePath_Absolute);
&MY_FILE.WriteLine(&strXml);
&MY_FILE.Close();

```

See Also

Enterprise PeopleTools 8.45 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 = SyncRequest(&MSG, Node.EAI);
&xmlResponseDoc = &MSG.GetXmlDoc();

```

See Also

Enterprise PeopleTools 8.45 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 a property, Cookies, of a Message object or as part of IBInfo. You can access this property only in an inbound synchronous response message or an outbound request message.

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(Message.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.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.Cookies = &SalesCookies;

/* Send the asynchronous request */
&SalesRequest.Publish();
```

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 the OnSend event.

Note. Properties set at the PeopleCode level take precedence over those set at the node, connector and transaction level.

IBInfo object	The IBInfo object is a member of the Message object. The methods and properties can be accessed by the IBInfo property on the Message object. You can use this object in publishing or synchronous request PeopleCode, or by adding an OnSend event to a message definition.
ConnectorInfo object	The ConnectorInfo object is a member of the IBInfo object. The methods and properties for this object can be accessed by the IBInfo property on the Message 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, or by adding an OnSend event to a message definition.
OnSend Event	The OnSend event is associated with a message definition and enables you to override target connector properties at runtime for a subscribing node transaction. This event 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 OnSend event to populate connector information in the publishing PeopleCode and then override it for a specific node.

Setting and Overriding Target Connector Properties Using the OnSend Event

You can use the OnSend event 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 the OnSend event to override connector properties at runtime, correct the PeopleCode in the OnSend event and resubmit the message.

Example: Setting Target Connector Properties Using the OnSend Event

The following example shows loading all connectors that exist for the node and adding one additional property, Filename.

```

Local Message &Msg;

//Get Message
&Msg = GetMessage();

&Bo = &MSG.IBInfo.LoadConnectorPropFromNode("nodename");

&Bo = &MSG.IBInfo.IBConnectorInfo.AddConnectorProperties
    ("FILENAME", "temp", %Property);
&MSG.IBInfo.ConnectorOverride = True;
ReturnToServer(&MSG);

```

Example: Overriding Connector Properties Using the OnSend Event

The following example demonstrates overriding target connector properties using the OnSend event for a rowset-based asynchronous message.

```

Local MESSAGE &MSG;
Local Boolean &bRet;
&MSG = GetMessage();

&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);

ReturnToServer(&MSG);

```

The following example demonstrates overriding target connector properties using the OnSend event for a nonrowset-based asynchronous message.

```

Local MESSAGE &MSG;
Local XmlDoc &xmldoc;
Local Boolean &bRet;
&MSG = GetMessage();

// 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);

ReturnToServer(&MSG);

```

See Also

Enterprise PeopleTools 8.45 PeopleBook: PeopleCode API Reference, “Message Class,” IBInfo Class

Enterprise PeopleTools 8.45 PeopleBook: PeopleCode API Reference, “Message Class,” ConnectorInfo Collection

Enterprise PeopleTools 8.45 PeopleBook: PeopleCode API Reference, “Message Class”

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 event that is associated with the message definition enables you to apply PeopleCode that determines whether the default local node accepts the inbound message.

Understanding Publication IDs and Subscription PeopleCode

Generally, if a message is successfully published, the PubID and PubNodeName properties are set to the publication ID and publishing system node name, respectively. The only exception is when a publish is performed as part of subscription PeopleCode. In this situation, the subscription process is always executed in deferred mode, due to performance considerations, and so the PubID is not populated.

Handling Inbound Asynchronous Transactions

Use the GetMessage or GetMessageXmlDoc built-in function and insert your PeopleCode in a message subscription.

Message Class Inbound Asynchronous Example 1

The following example uses the GetMessage built-in function:

```
Local Message &MSG;
Local Rowset &LEVEL0;
Local Record &SALES_ORDER_INFO, &REC;

&MSG = GetMessage();
&SALES_ORDER_INFO = CreateRecord(Record.QE_SALES_ORDER);

&LEVEL0 = &MSG.GetRowset();
&REC = &LEVEL0(1).QE_SALES_ORDER;
&REC.CopyFieldsTo(&SALES_ORDER_INFO);

SQLEXec("DELETE FROM PS_QE_SALES_ORDER");

&SALES_ORDER_INFO.Insert();
```

Message Class Inbound Asynchronous Example 2

The following example shows subscription 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:

```

/* Simple PeopleCode Subscribe- - Check the PSCAMA*/
Local Message &MSG;
Local Rowset &MSG_RS;
Local Record &REC_NAME_PREFIX, &REC, &REC_RL;

&MSG = GetMessage();
&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;
End-If;

    /*****
    /***** 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;
End-For;

```

Message Class Inbound Asynchronous Example 3

The following example shows subscription PeopleCode with multiple transactions:

```

Local Message &STOCK_MSG;
Local Rowset &HDR_RS, &LN_RS;

```

```

Local Record &HDR_REC, &hdr_rec_msg, &LN_REC, &LN_REC_MSG;

/*
This subscription 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)
*/

/* Get the STOCK_MSG App. Message rowset */
&STOCK_MSG = GetMessage();

/* 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 = &STOCK_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 */
    &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;

```


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 &STOCK_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 message is built.
This references MSR_HDR_INV (Header) and DEMAND_INF_INV (Line) records */

/* Create an instance of the STOCK_REQUEST message */
&STOCK_MSG = CreateMessage(Message.STOCK_REQUEST);

/* Create an App. Message Rowset that includes the
MSR_HDR_INV (Header) and DEMAND_INF_INV (Line)*/
&hdr_rs = &STOCK_MSG.GetRowset();

/* Create a SQL object to select the Header rows */
&hdr_sql = CreateSQL("Select * from PS_MSR_HDR_INV
    WHERE BUSINESS_UNIT='M04A1'
    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 &STOCK_MSG.Size > %MaxMessageSize Then
        &STOCK_MSG.Publish();
        &I = 1;
        /* Create a new instance of the message object */
        &STOCK_MSG = CreateMessage(Message.STOCK_REQUEST);
        &hdr_rs = &STOCK_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
    /* &STOCK_MSG message structure */
    &hdr_rec.CopyFieldsTo(&hdr_rec_msg);

    /* Publish the last message if it has been changed*/
    If &hdr_rec_msg.IsChanged Then
        &STOCK_MSG.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 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, &datetime;
Local XmlDoc &xmldoc;
Local string &data;
Local boolean &bo;

&CRLF = Char(13) | Char(10);

&MSG = GetMessage();

/* this is how one would access data from IBinfo and
/* IBConnectorInfo objects*/
&bo = &MSG.IBInfo.ConnectorOverride;

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

    &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();

&data0 = &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);

/* 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.DESCLONG).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 Message &MSG;
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 &bo;

&MSG = GetMessage();

/* this is how one would access data from IBInfo and
/* IBConnectorInfo objects*/

&bo = &MSG.IBInfo.ConnectorOverride;

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

    &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;

&xmldoc = &MSG.GetXmlDoc();

&CRLF = Char(13) | Char(10);

/* get the content data from the message into an xmldoc object*/
&xmldoc = GetMessageXmlDoc();

&root = &xmldoc.DocumentElement;

/* Get values out of XmlDocument */
&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");
&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 message subscription using a component interface in the subscription. This example shows error trapping and has multilanguage support:

```

PanelGroup 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;
        &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;

```

```

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();

    /*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 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 = GetMessage();
&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 GetMessageXmlDoc built-in function:

```

Local XmlDoc &BIGMAN;
Local XmlNode &node, &root;
Local string &outstring;
Local Rowset &LEVEL0;
Local Record &SALES_ORDER_INFO, &REC;

&CRLF = Char(13) | Char(10);

&BIGMAN = GetMessageXmlDoc();

&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

Use the GetMessage or GetMessageXmlDoc built-in function and place your PeopleCode in the message OnRequest event.

When you receive an inbound request message, the response message must conform to the return type that is expected by the sending code. If the sender uses the SyncRequest method in the PeopleCode Message class, you must respond with a Message object. If the sender uses the SyncRequestXmlDoc built-in function, you must respond with an XmlDoc or SoapDoc object.

Message Class Inbound Synchronous Example

The following example uses the GetMessage built-in function:

```

Local Message &request, &response;

&request = GetMessage();
&response = CreateMessage(Message.QE_SO_SYNC_RESP);
&response.GetRowset().GetRow(1).GetRecord(Record.QE_SO_RESPONSE)
    .GetField(Field.DESCRLONG).Value = "He's a nice person ";

ReturnToServer(&response);

```

XmlDoc Class Inbound Synchronous Example

The following example uses the GetMessageXmlDoc built-in function:

```

Local XmlDoc &xmlRequestDoc;
Local XmlDoc &xmlResponseDoc;
Local XmlNode &select;
Local array of XmlNode &whereClause;

Local string &recordName;
Local string &fieldName;
Local string &fieldValue;
Local Rowset &outputRowset;
Local boolean &b;

&xmlRequestDoc = GetMessageXmlDoc();
&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;

&xmlResponseDoc = CreateXmlDoc();
&b = &xmlResponseDoc.CopyRowset(&outputRowset);
ReturnToServer(&xmlResponseDoc);

```

SoapDoc Class Inbound Synchronous Example

The following example uses the GetMessageXmlDoc built-in function:

Note. Because GetMessageXmlDoc is an XmlDoc built-in function, you must receive the inbound request message as an XmlDoc object, then convert it to a SoapDoc object to process it with SOAP methods. Likewise, because the request was sent by using SyncRequestXmlDoc, you must convert the SoapDoc response message to an XmlDoc object before returning it.

```

Local XmlDoc &request, &response;
Local string &strXml;
Local SoapDocs &soapReq, &soapRes;

&soapReq = CreateSoapDoc();

&request = GetMessageXmlDoc();
&soapReq.XmlDoc = &request;
&OK = &soapReq.ValidateSoapDoc();
&parmN = &soapReq.GetParmName(1);
&parmV = &soapReq.GetParmValue(1);

&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;
ReturnToServer(&response);

```

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 `InboundPublish` method, a method of the `Message` object, to work with rowset-based and nonrowset-based messages. Use the `InboundPublishXMLDoc` function to work with nonrowset-based messages.

The following example shows an inbound publish off a subscription with a rowset-based message:

```
Local Message &MSG, &MSG_REMOTE;
Local Rowset &rs;

&MSG = GetMessage();
&rs = &MSG.GetRowset();
/*create the message to be re-published from a simulated remote node*/

&MSG_REMOTE = CreateMessage(Message.QE_FLIGHTPLAN);

&MSG_REMOTE.CopyRowset(&rs);

&MSG_REMOTE.InBoundPublish(Node.REMOTE_NODE);
```

The following example shows an inbound publish off a subscription with a nonrowset-based message:

```
Local Message &MSG, &MSG_REMOTE;
Local XmlDocument &xmldoc;
Local Rowset &rs;

&MSG = GetMessage();
&rs = &MSG.GetRowset();
/*create the message to be re-published from a simulated remote node*/

&MSG_REMOTE = CreateMessage(Message.QE_FLIGHTPLAN);

/* populate the &xmldoc with data to be re-published*/

&MSG_REMOTE.SetXmlDoc(&xmldoc);

&MSG_REMOTE.InBoundPublish(Node.REMOTE_NODE);
```

The following examples shows using `InboundPublishXmlDoc` off a subscription with a nonrowset-based message.

```
Local XmlDocument &xmldoc;
Local boolean &bReturn;

&xmldoc = GetMessageXmlDoc();

&bReturn = InBoundPublishXmlDoc(&xmldoc, Message.QE_FLIGHTPLAN, 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 8.45 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 8.45 PeopleBook: PeopleCode API Reference*, "SOAPDoc Class".

Message Class Validation

Have a message receiving process validate incoming data by:

- Selecting the Validate box on the Message Properties page of the message in PeopleSoft Application Designer. This option is for messages with defined records and fields only.
- Using the ExecuteEdits method on the message 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
- %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 subscription 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.

Exit()

In the Exit() form, 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.
- Commits the transaction.
- Stops processing.

Following is an example of the Exit() form:

```
&MSG.ExecuteEdits();
If &MSG.IsEditError then
    App_Specific_Error_Processing();
    Exit();
Else
    Specific_Message_Processing();
```

```
End-if;
```

Exit(1)

In this form, the Exit built-in 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 the Exit(1) form:

```
&MSG.ExecuteEdits();
If &MSG.IsEditError then
    Exit(1);
Else
    Process_Message();
End-if;
```

See Also

[Chapter 14, “Using Integration Broker Monitor,” page 291](#)

Enterprise PeopleTools 8.45 PeopleBook: PeopleCode Language Reference,
“PeopleCode Built-in Functions,” Exit

Correcting Messaging Errors

If you anticipate few errors, correct them on the Message Errors page in the Integration Broker Monitor.

If you have many errors, you may want to create a page that a user can use to correct the data for a particular message. Generally, users will be correcting errors that were detected when running the ExecuteEdits method in the PeopleCode Message class or another built-in function as part of the receiving code.

For the search record for your page, you can create a view based on the MSGNAME and SUBCONSTATUS fields of the PSAPMSGSUBCON record. MSGNAME lists all of the messages that have subscription contracts. The SUBCONSTATUS field has the following values:

Status	Value
ERROR	0
NEW	1
STARTED	2
WORKING	3

Status	Value
DONE	4
RETRY	5
TIMEOUT	6
EDITED	7
CANCELLED	8

If you create a page, you must specify it in the message definition properties. After you define it, you can either use it as a regular page or access it from the Message Details link in the Integration Broker Monitor.

Correcting Message Errors in PeopleCode

The following code is located in the PreBuild event of a page that's used to fix message errors. It gets the inbound message as specified by the search record (PSAPMSGSUBCON) and loads the fields of a work record (MS_EDIT_WRK) with the data from the XMLDoc &QE_PLAYER:.

```
Local XmlNode &player_id_node, &root, &player_status_node, &desc_node;

&QE_PLAYER = GetSubXmlDoc(PSAPMSGSUBCON.PUBID, PSAPMSGSUBCON.PUBNODE,
    PSAPMSGSUBCON.CHNLNAME, PSAPMSGSUBCON.APMGVER, PSAPMSGSUBCON.MSGNAME);

&root = &QE_PLAYER.DocumentElement;

/* Get values out of XMLDoc */
&node_array = &root.GetElementsByTagName("PLAYER ID");
&acct_id_node = &node_array.Get(1);
MSG_EDIT_WRK.QE_PLAYER_ID = &player_id_node.NodeValue;

&node_array = &root.GetElementsByTagName("PLAYER STATUS");
&player_status_node = &node_array.Get(1);
MSG_EDIT_WRK.QE_PLAYER_STATUS = &player_status_node.NodeValue;
&node_array = &root.GetElementsByTagName("DESC");
&desc_node = &node_array.Get(1);
MSG_EDIT_WRK.DESCRLONG = &desc_node.NodeValue;
```

The following code is located in the SavePostChange event. It takes the values from the work record and repopulates the rowset. Because the data that is in the rowset is identical to the data in the message, this changes the values of the fields in the message. When you use the Update method, the status of the message in the Integration Broker Monitor is automatically changed to Edited.

```
Local Rowset &RS_PLAYER;
Local Message &MSG_EDIT_TEST;

&MSG_EDIT_TEST = GetSubMessage(PSAPMSGSUBCON.PUBID, PSAPMSGSUBCON.PUBNODE,
    PSAPMSGSUBCON.CHNLNAME, PSAPMSGSUBCON.MSGNAME, PSAPMSGSUBCON.SUBNAME);
```



```

&RS_PLAYER = &MSG_EDIT_TEST.GetRowset();

&RS_PLAYER(1).QE_PLAYER.QE_PLAYER_ID.Value = MSG_EDIT_WRK.QE_PLAYER_ID;
&RS_PLAYER(1).QE_PLAYER.QE_PLAYER_STATUS.Value = MSG_EDIT_WRK.QE_PLAYER_STATUS;
&RS_PLAYER(1).QE_PLAYER.DESCRLONG.Value = MSG_EDIT_WRK.DESCRLONG;

&MSG_EDIT_TEST.Update();

```

Correcting Data Errors on Pages

To use a predefined page to correct data errors, specify the page on the Use tab in the Message Properties dialog box.

To open the page, open the Integration Broker Monitor and select the Sub Contracts page. Select the message in error and click Message Details.

See Also

[Chapter 14, “Using Integration Broker Monitor,” page 291](#)

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 properties and methods within the Message object. For nonrowset-based XML messages, you could not use the Message object, 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.

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 built-in functions, SetXMLDoc and GetXMLDoc.

SetXMLDoc Use this built-in function to load and pass nonrowset-based data into the Message object.

GetXMLDoc Use this built-in function to get nonrowset-based data out of the message object.

Using the SetXMLDoc Built-In Function

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(Message.QE_F18_ASYNC_XMLDOC);
// Load the Message object with the xmldoc data.
&MSG.SetXmlDoc(&XmlDoc);

```

```
// perform a publish for the nonrowset-based message
&MSG.Publish();
```

Using the GetXMLDoc Built-In Function

The following code example shows how to use GetXMLDoc to get nonrowset-based XML out of the Message object.

```
Local MESSAGE &MSG;
Local XMLDOC &XmlDoc;

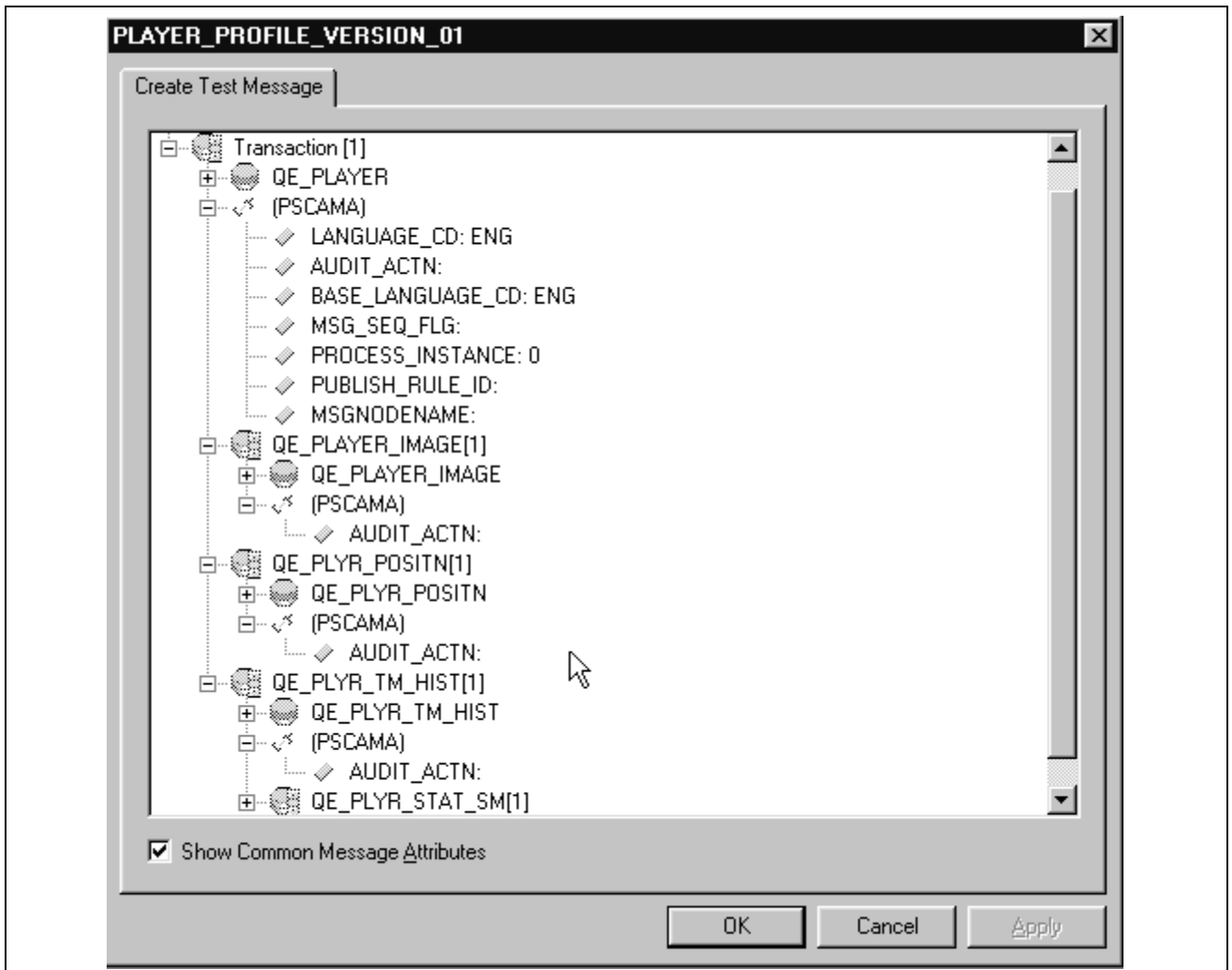
// Get the Message object
&MSG = GetMessage();
// get an xmldoc object loaded with the content data.
&XmlDoc = &MSG.GetXmlDoc();
```

See Also

Enterprise PeopleTools 8.45 PeopleBook: PeopleCode Language Reference

Generating Test Messages

Because the message is not actually sent, creation of a test message is not a full send-and-receive test; however, it provides useful information about the message definition and its associated PeopleCode.



Creating a test message

To create a test message:

1. In the message definition, right-click the message version that you want to test and select Create Test Message.

A test message appears showing the structure of the message. Select the Show Common Message Attributes box to display the PeopleSoft Common Application Message Attributes (PSCAMA) fields in the test message.

2. Enter sample data for the fields in the message.

To enter sample data for a field, double-click the field and enter a value. You can also right-click the field and select Edit Value.

Note. To publish every field in the message, you must enter sample data in at least one field of every record.

3. Click OK to publish the test message.
4. Use Integration Broker Monitor to view the results of the test message.

See Also

Chapter 5, “Understanding Supported Message Structures,” PSCAMA , page 70

Chapter 14, “Using Integration Broker Monitor,” page 291

CHAPTER 9

Configuring Nodes and Transactions

This chapter discusses how to:

- Configure nodes.
- Configure transactions.

Understanding Configuring Nodes and Transactions

When an integration engine receives a message, it searches the transaction definitions associated with that message definition to determine the valid target nodes for the message, then passes the message to the synchronous or asynchronous request processes. The handler directs the message to each valid node, using the gateway and target connector specified by the node definition. It could also direct the message to local Subscription or OnRequest PeopleCode. A transaction might also be modified by a defined relationship, which could involve a transformation.

Prerequisites

To configure a node and its associated transactions, at least one gateway with one connector must be defined.

Routing Types

When you configure a node or override transaction properties, you must also specify the routing type.

The routing type specifies how the integration engine should determine whether to send a message to the selected node.

This property is ignored for inbound transactions.

Explicit Routing

When you select explicit routing, the current node won't be included as a target node unless you specify it directly in your sending PeopleCode. You can either:

- Specify the target node as a parameter of the message class Publish or SyncRequest methods.
- Specify the target node as a parameter of the PublishXMLDoc or SyncRequestXMLDoc built-in functions.

If you selected a node type of *External*, the routing type is *Explicit* by default.

Implicit Routing

When you select implicit routing, all nodes with this routing type are included as target nodes unless the sending PeopleCode references specific nodes. In that case, only the referenced nodes are targets. If you selected a node type of *PIA*, the routing type is *Implicit* by default.

Note. You can also include the selected node — or filter the list of included nodes to produce a single target node — using the OnRouteSend PeopleCode event associated with the message definition.

See [Chapter 8, “Sending and Receiving Messages,” Routing Events, page 156.](#)

Configuring Nodes

This section discusses how to:

- Define node.
- Specify contact information.
- Define node properties.
- Specify node gateways and connectors.
- Rename or delete node definitions.

Understanding Configuring Nodes

To configure nodes you use the Node Definitions page in the Node Definitions component (IB_NODE). To access the Node Definitions page, select PeopleTools, Integration Broker, Node Definitions.

Understanding 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)

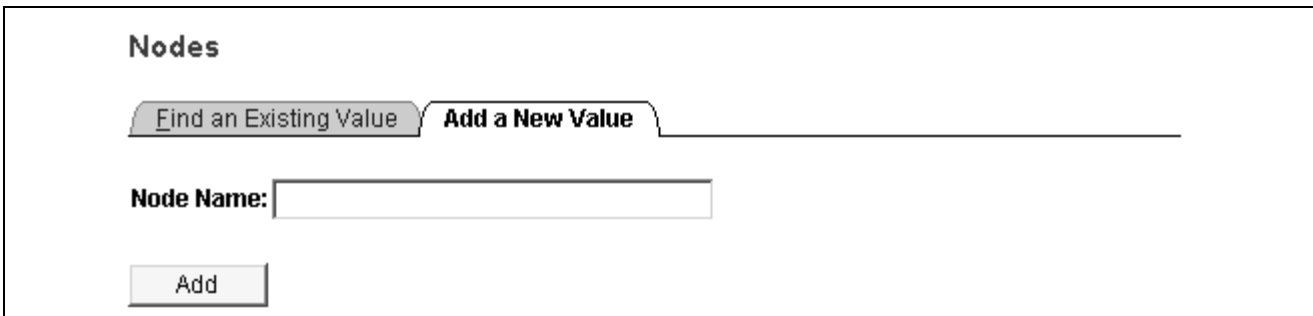
Note. In practice, only portals use nodes designated simply as *Local*. The only local node definition used by Integration Broker is the one designated *Default Local*, which represents the database you're signed on to.

Creating Node Definitions

This section describes how add new nodes to the system or select existing nodes, and define node parameters.

Adding New Nodes

To define a new node, select PeopleTools, Integration Broker, Node Definitions.



The screenshot shows a web interface titled "Nodes". It has two tabs: "Find an Existing Value" and "Add a New Value". The "Add a New Value" tab is active. Below the tabs, there is a label "Node Name:" followed by a text input field. At the bottom of the form is an "Add" button.

Adding a new node

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, 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.

Selecting Existing Nodes

Select PeopleTools, Integration Broker, Node Definitions. Click the Search button to select from a list of defined nodes. Click the node with which to work. The Node Definitions tab displays.

Defining Node Parameters

To define node parameters, use the Node Definitions tab in the Node Definitions component.

Node Definitions

Contacts

Properties

Connectors

Transactions

Portal

Node Name TEST

Details

*Description

Default Local Node

No

☐ Local Node

☒ Active Node

☐ Non-Repudiation

*Node Type

PIA

*Routing Type

Implicit

*Authentication Option

None

Hub Node

Master Node

Company ID

Image Name

Code Set Group Name

Node Definitions — Node Definitions tab

Description	Enter a descriptive name for the node.
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 9, "Configuring Nodes and Transactions," Renaming or Deleting Node Definitions, page 203.
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. If you clear this check box to deactivate the node, transactions and relationships that use it are also deactivated. Activating the node won't automatically reactivate the transactions and relationships; they must be individually reactivated. You can't deactivate the default local node.

See [Chapter 9, “Configuring Nodes and Transactions,” Editing Transaction Details, page 209](#) and [Chapter 13, “Administering Relationships,” page 273](#).

Non-Repudiation

Select to activate nonrepudiation for the current node.

Note. You must also activate nonrepudiation in the definition of each message for which you want this feature.

See [Chapter 3, “Administering Messaging Servers for Asynchronous Messaging,” page 21](#).

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.

Routing Type

Specify how the integration engine should determine whether to send a message to the current node.

After you create a node, the routing type for the node and the routing type for outbound transactions function independently. For example, if you create a node and choose a routing type of *Explicit*, and then create several outbound synchronous and outbound asynchronous transactions and save your work, the routing type for the transactions you created defaults to *Explicit*. However, if you later change the node routing type, the routing type does not automatically get changed for any of the existing outbound transactions. You must manually change the transaction routing type, if desired, on the Transactions Detail tab for the transaction.

See [Chapter 9, “Configuring Nodes and Transactions,” Routing Types, page 195](#).

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 message, 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. Messages 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.

Important! Single signon isn't 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 messages 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.

Password

Displays when the Authentication Option is *Password*.

Enter the node password.

Confirm Password

Reenter the node password you entered in the Password field.

Hub Node

Select the name of a node that will serve as a “gatekeeper” for the current node. All transactions outbound from the default local node to the current node will be redirected to the selected hub node instead. You can select any existing PeopleSoft Pure Internet Architecture node for this purpose. You must define appropriate transactions and relationships at that node to route the message properly.

Use this field to add hub capability for existing transactions that specify the current node as a target, without having to modify those transactions.

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.

See [Chapter 13, “Administering Relationships,” page 273](#).

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.

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 the default local node use this association to search for message data requiring translation.

See [Chapter 12, “Applying Filtering, Transformation and Translation,” Performing Data Translation, page 259](#).

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.

All transactions associated with this node definition are copied as well, so the messaging interactions you've developed with one node can be duplicated for another node in a single step.

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.

See [Chapter 9, “Configuring Nodes and Transactions,” Renaming or Deleting Node Definitions, page 203.](#)

Specifying Contact Information

Select the Contacts tab to access the Node Definitions – Contacts 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 URL	The address of the Contact Manager's support web site, if there is one.

Defining Node Properties

Select the Properties tab to access the Node Definitions – 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: <i>Category:</i> The property is used for categorization. <i>Ident:</i> The property is used for identification. <i>Search:</i> The property is used for searching.
Property Name	Enter a new property name or select an existing property of the selected name type.

Specifying Gateways and Connectors

Select the Connectors tab to access the Node Definitions – Connectors page.

The screenshot shows the 'Node Definitions – Connectors' page. At the top, there are tabs for 'Node Definitions', 'Contacts', 'Properties', 'Connectors' (selected), 'Transactions', and 'Portal'. Below the tabs, the 'Node Name' is 'QE_LOCAL'. Under the 'Details' section, there are two input fields: 'Gateway ID' with the value 'LOCAL' and 'Connector ID' with the value 'HTTPTARGET'. Below this is the 'Properties' section, which has a table with three columns: '*Property ID', '*Property Name', and 'Required Value'. The table contains three rows: 1. 'HEADER' with 'sendUncompressed', 2. 'HTTPPROPERTY' with 'Method', and 3. 'PRIMARYURL' with 'URL'. Each row has a search icon and a plus/minus icon. At the bottom of the 'Properties' section is a button labeled 'Password Encryption Utility'.

Node Definitions — Connectors page: Properties tab

Use this page to specify how the default local node should send messages to the current node.

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.

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.

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 4, “Managing Integration Gateways,” Editing Connector Properties, page 39](#).

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; 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.

See Also

[Chapter 9, “Configuring Nodes and Transactions,” Editing Transaction Details, page 209](#)

[Chapter 4, “Managing Integration Gateways,” Encrypting Passwords, page 44](#)

[Chapter 4, “Managing Integration Gateways,” Editing Connector Properties, page 39](#)

Renaming or Deleting Node Definitions

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 a relationship that refers to that node — by changing the reference to use the new name, or by deactivating the relationship if the node is deleted.

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 a node that hosts a portal. As a result, the Delete Node button is hidden on a node definition if the node hosts a portal.

Note. If you upgraded your PeopleSoft application from a PeopleTools 8.1x release, the newly created default local node definition must be renamed, so you must first remove any remaining live message data if you didn't do so before the upgrade.

Renaming or deleting a node requires the following actions:

1. Deactivate all the domains in your messaging system.
 - a. In the PeopleSoft Pure Internet Architecture, select PeopleTools, Integration Broker, Monitor, Monitor Message, Domain Status.
 - b. On the Domain Status page, select All Domains 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 Message Details or Synchronous Details component.
- You can archive messages with a batch process using the Archive Messages component.
- You can purge message data using one of several Data Mover scripts delivered with PeopleSoft Integration Broker. You'll find them in *PS_HOME*\scripts:

AppMsgPurgeLive.dms	Deletes the queue data from every live message table in the database.
AppMsgPurgeAll.dms	Deletes the message data from every live message table and every archive message table in the database. This is the recommended procedure when upgrading from earlier versions of PeopleTools, because the archived data is largely incompatible with the new release.

3. Rename or delete the desired node definition.
4. Reboot the web server.
5. Reactivate the messaging domains.
 - a. In the PeopleSoft Pure Internet Architecture, select PeopleTools, Integration Broker, Monitor, Monitor Message, Domain Status.
 - 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 9, “Configuring Nodes and Transactions,” Renaming or Deleting Node Definitions, page 203](#)

[Chapter 14, “Using Integration Broker Monitor,” Monitoring Asynchronous Message Details, page 336](#)

[Chapter 14, “Using Integration Broker Monitor,” Viewing Synchronous Message Details, page 341](#)

[Chapter 14, “Using Integration Broker Monitor,” Monitoring Messaging System Information, page 304](#)

[Chapter 14, “Using Integration Broker Monitor,” Purging Messaging Tables, page 348](#)

Configuring Transactions

This section discusses how to:

- View available transactions.

- Define a transaction.
- Edit transaction details.
- Edit transaction message details.
- Edit transaction connector properties.

Understanding Transactions

A transaction is the basic unit of work in an integration. Each transaction is associated with a specific node. You use a transaction to designate:

- A message that the current node can send or receive.
- Whether the message is inbound or outbound.
- Whether the message is synchronously or asynchronously transmitted.

Thus there are four distinct transaction types:

- Outbound asynchronous (OutAsync).
- Outbound synchronous (OutSync).
- Inbound asynchronous (InAsync).
- Inbound synchronous (InSync).

Note. When you define a node or transaction, *current* refers to the definition you have open — for example, the *current node*. *Selected* refers to an item you’ve indicated other than the definition you have open — for example, the *selected transaction*.

See [Chapter 2, “Understanding PeopleSoft Integration Broker,” Transaction Types, page 10](#).

Transaction Definitions for Cross-Node Messaging

Each PeopleSoft Integration Broker node involved in an integration point must define a transaction for each node with which it communicates. For example, for an integration in which node A transmits message X to both node B and node C:

- In the node A database, the NODE_B definition must contain an *outbound* transaction that specifies request message MSG_X.
- In the node A database, the NODE_C definition must contain an *outbound* transaction that specifies request message MSG_X.
- In the node B database, the NODE_A definition must contain an *inbound* transaction that specifies request message MSG_X.
- In the node C database, the NODE_A definition must contain an *inbound* transaction that specifies request message MSG_X.

For this integration, transactions don’t need to be defined for the default local node in any database.

Note. For a basic integration, all transactions involved must be of the same transmission type — all synchronous or all asynchronous. However, you can mix transmission types by defining transaction modifiers.

Transaction Definitions for Local Messaging

Because a local integration involves only the default local node, it requires only a single transaction. For example, for an integration in which node A transmits message X to itself (a local transaction), the NODE_A default local node definition in the node A database must contain an *outbound* transaction that specifies request message MSG_X.

No other transaction is necessary. Because NODE_A is the default local node, the inbound transaction is implicit, and the integration engine automatically routes it appropriately.

See Also

[Chapter 13, “Administering Relationships,” page 273](#)

Common Elements Used in This Section

Status

Select from:

Active: This transaction is considered for use by the request handlers.

Inactive: This transaction isn’t considered for use by the request handlers.

Return to Transaction List

Select Return to Transaction List, and the Transactions page for the selected node appears.

Important! Save the transaction before returning to the transaction list, or your changes — including the creation of a new transaction — are canceled.

Viewing Transactions

Select PeopleTools, Integration Broker, Node Definitions and select the Transactions tab to access the Node Definitions — Transactions page.

Node Name: PT_LOCAL

Transactions						
Edit	Transaction Type	Request Message	Request Message Version	Effective Date	Status	Delete
1 Edit	InAsync	IB_INST_VER_ASYNC_MSG	VERSION_1	03/13/2002	Active	-
2 Edit	InAsync	IB_INST_VER_RESP_MSG	VERSION_1	03/13/2002	Inactive	-
3 Edit	InSync	IB_INST_VER_SYNC_MSG	VERSION_1	03/13/2002	Active	-
4 Edit	OutAsync	IB_INST_VER_ASYNC_MSG	VERSION_1	03/13/2002	Inactive	-
5 Edit	OutAsync	IB_INST_VER_RESP_MSG	VERSION_1	03/13/2002	Active	-
6 Edit	OutSync	IB_INST_VER_SYNC_MSG	VERSION_1	03/13/2002	Inactive	-
7 Edit	InAsync	EMAIL_MSG	VERSION_1	01/01/1900	Active	-
8 Edit	InAsync	ROLESYNCH_MSG	VERSION_1	01/01/1900	Active	-
9 Edit	OutAsync	EMAIL_MSG	VERSION_1	01/01/1900	Active	-
10 Edit	OutAsync	ROLESYNCH_MSG	VERSION_1	01/01/1900	Active	-

[Add Transaction](#) [Copy All Transactions](#)

Node Definitions — Transactions page

Edit	Click to access the Transaction Detail page and modify the properties of the transaction.
Add Transaction	Click to access the Node Transactions page and define a new transaction.
Copy All Transactions	Click to copy all of the current node's transactions to another node definition. The properties of each copied transaction are identical to the originals except for the node name, which changes to reflect the node definition to which you copied the transactions.

This page lists all the transactions associated with the current node, including the transaction type, the request message and version, and the effective date for each transaction.

Note. If you delete a transaction row, any transaction modifiers that use that transaction are also deleted.

Note. The transactions listed on this page always define interactions between the current node and the default local node (the database you're signed on to). If the current node is the default local node, the transactions are local. The transaction direction is always with respect to the default local node — outbound is *from* the default local node, and inbound is *to* the default local node.

See Also

[Chapter 9, “Configuring Nodes and Transactions,” Defining Transactions, page 207](#)

[Chapter 9, “Configuring Nodes and Transactions,” Editing Transaction Details, page 209](#)

[Chapter 13, “Administering Relationships,” Managing Transaction Modifiers, page 281](#)

Defining Transactions

Click Add Transaction on the Node Definitions — Transactions page.

The screenshot shows the 'Node Transactions' page with the 'Add a New Value' tab selected. The form contains the following fields:

- Node Name:** A text input field containing 'QE_LOCAL' with a search icon on the right.
- Effective Date:** A date input field showing '01/16/2004' with a calendar icon on the right.
- Transaction Type:** A dropdown menu with 'Outbound Synchronous' selected.
- Request Message:** A text input field containing 'QE_SYNC_LOCAL_MSG' with a search icon on the right.
- Request Message Version:** A text input field containing 'VERSION_1' with a search icon on the right.

At the bottom left of the form is an 'Add' button.

Node Transactions page

Node Name	Select the name of a node defined in the current database. The name of the current node is already entered, but you can select a different node with which to associate this transaction.

Note. If you select a different node, the selected node’s definition opens when you save the transaction.

Effective Date

Enter the date which the message is to become effective.

Transaction Type

Select from:

Outbound Asynchronous (OutAsync): The default local node sends a request message to the selected node.

Outbound Synchronous (OutSync): The default local node sends a request message to the selected node, requiring a response.

Inbound Synchronous (InSync): The default local node receives a request message from the selected node, requiring a response.

Note. Inbound transactions aren’t supported for the default local node definition.

Request Message

Select the message to be sent or received with this transaction.

Note. The PeopleCode associated with the message must support the selected transaction type.

Request Message Version

Select the version of the request message to be sent or received with this transaction.

Add

Click to confirm the parameters you entered for the new transaction. The Transaction Detail page appears.

Note. To edit an existing transaction definition, select the Find an Existing Value tab and search for a transaction based on any of its key parameters. If you select a transaction associated with a different node than the current one, the selected node’s definition opens when you save your changes.

No Multiple Synchronous Targets

In practice, only one target node at a time can receive a 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 in one of the following ways:

- Specify the target node in your sending PeopleCode as a parameter of the Message class SyncRequest method. This forces Integration Broker to consider only the transactions defined for that node.
- Specify the target node in your sending PeopleCode as a parameter of the SyncRequestXMLDoc built-in function. This forces Integration Broker to consider only the transactions defined for that node.
- Filter the available target nodes in your OnRouteSend PeopleCode.

If you specify a target node in your sending PeopleCode, *no* other nodes are considered valid targets, including those with a routing type of *Implicit*.

See Also

Enterprise PeopleTools 8.45 PeopleBook: PeopleCode API Reference, “Message Class”

Enterprise PeopleTools 8.45 PeopleBook: PeopleCode API Reference, “XmlDoc Class”

Editing Transaction Details

To access the Transaction Detail page, click Edit on the Node Definitions – Transactions page or click Add on the Node Transactions – Add a New Value page.

The screenshot shows the 'Transaction Detail' page for a node named 'QE_LOCAL'. The page has three tabs: 'Transactions' (selected), 'Messages', and 'Connectors'. The 'Transaction Detail' section contains the following fields:

- Effective Date:** 01/16/2004
- Transaction Type:** OutSync
- Request Message:** QE_SYNC_LOCAL_MSG
- Request Message Version:** VERSION_1
- *Status:** Active (dropdown menu)
- Routing Type:** Implicit (dropdown menu)
- Override Connector:** ☒
- Gateway ID:** LOCAL (text input with search icon)
- Connector:** HTTPTARGET (text input with search icon)
- Comment:** (text input with up/down arrows)

A link 'Return to Transaction List' is located at the bottom right of the form.

Transaction Detail page

Note. The transaction's basic parameters can be altered only by defining a new transaction with different values.

Status	Determines if the transaction is active. The valid values are: • Inactive • Active
Routing Type (Outbound transactions only)	Override the selected node's defined routing type, which is the default. After you create a node, the routing type for the node and the routing type for outbound transactions function independently. For example, if you create a node and choose a routing type of <i>Explicit</i>, and then create several outbound synchronous and outbound asynchronous transactions and save your work, the routing type for the transactions you created defaults to <i>Explicit</i>. However, if you later change the node routing type, the routing type does not automatically get changed for any of the existing outbound transactions. You must manually change the transaction routing type, if desired, on the Transactions Detail tab for the transaction.

**Override
Connector**(Outbound
transactions only)

Select to choose a gateway and target connector for the current transaction that overrides those specified for the selected node. The Gateway ID and Connector fields appear. You must select a gateway ID and a connector.

Note. A connector override doesn't need to specify a different gateway or connector than the one specified by the node; it might just apply different properties for the same connector.

Connector

Select the connector you want the current transaction to use from the list of connectors registered with the selected gateway. When you save the transaction, a new page appears where you can override the connector's properties.

See [Chapter 9, "Configuring Nodes and Transactions," Editing Transaction Connector Properties, page 212.](#)

Activating Transactions

When a transaction is active, PeopleSoft Integration Broker can immediately use it to route messages to or from its associated node. However, messaging also depends on the status of the messages and the node — their definitions in the local database must all be set to active status. The following table outlines how this dependency is enforced.

Condition	Action	Dependency
Message is active.	Deactivate the message.	All associated transactions are deactivated.
Message is active.	Create a new transaction.	Transaction is active by default.
Message is inactive.	Create a new transaction.	Transaction is inactive by default.
Message is inactive.	Activate a transaction.	Allowed, with a warning.
Message is inactive.	Activate the message.	No dependency.
Node is active.	Deactivate the node.	All associated transactions are deactivated.
Node is active.	Create a new transaction.	Transaction status matches message status.
Node is inactive.	Create a new transaction.	Transaction is inactive by default.
Node is inactive.	Activate a transaction.	Not allowed.
Node is inactive.	Activate the node.	No dependency.

Condition	Action	Dependency
Message and node are active.	Activate a transaction.	Allowed.
Message and node are any status.	Deactivate a transaction.	No dependency.

See Also

[Chapter 9, “Configuring Nodes and Transactions,” Routing Types, page 195](#)

[Chapter 9, “Configuring Nodes and Transactions,” Editing Transaction Connector Properties, page 212](#)

Editing Transaction Message Details

Select the Messages tab on the Transaction Detail page to access the Messages page.

Transactions Messages Connectors

Node Name QE_LOCAL

Transaction Messages Find | View All First 1 of 1 Last

Effective Date 01/16/2004

Status Active

Transaction Type OutSync

*Synchronous Logging Header and Detail

Request Message

Message Name QE_SYNC_LOCAL_MSG

Message Version VERSION_1

External Name

Response Message

Message Name QE_SYNC_LOCAL_MSG

Message Version VERSION_1

External Name

[Return to Transaction List](#)

Messages page

Note. The Synchronous Logging drop-down list and the Response Message group box appear only if you defined the transaction as synchronous.

Synchronous Logging	This option enables you to set select the level of information logging for synchronous messages. The valid values are: • <i>Header Only</i>. Log header information only. With this option, you can view synchronous message header information on the Integration Broker Monitor Synchronous Details page. • <i>Header and Detail</i>. Log header and message detail information. With this option, you can view synchronous message header information and XML message content on the Integration Broker Monitor Synchronous Details page. • <i>No Logging</i> (Default). Turn off all logging.
External Name	The name of the request or response message as defined in the default local node’s database might not match the name by which it’s defined in the selected node’s database. Enter an external name for the message that the selected node recognizes. The external name applies to the message only when it’s sent or received by the selected node using this transaction. This way each participating node handles the message using the name it recognizes.
Message Name	Select the response message to be sent or received by the selected node. Note. This message must be assigned to the same channel as the request message.
Message Version	Select the version of the response message that the selected node will use in this transaction.

See Also

[Chapter 14, “Using Integration Broker Monitor,” page 291](#)

Editing Transaction Connector Properties

Select the Connectors tab on the Transaction Detail page to access the Connectors page.

Transactions
Messages
Connectors

Node Name QE_LOCAL

Transaction Connectors

Message Name QE_SYNC_LOCAL_MSG
Effective Date 01/16/2004

Gateway ID LOCAL
*Status Active

Connector ID HTTPTARGET
Transaction Type OutSync

Properties
Customize | Find |
First 1-3 of 3 Last

	*Property ID	*Property Name	Value
1	HEADER	sendUncompressed	Y
2	HTTPPROPE	Method	POST
3	PRIMARYURI	URL	

Password Encryption Utility

[Return to Transaction List](#)

Connectors page

Note. This page appears only if you specified a connector override on the Transaction Detail page and saved your changes.

Because the selected gateway and target connector override those specified for the selected node, you can also override the connector's properties. These fields are identical to the Node Definitions — Connectors page.

If any of the new connector properties require a password, click the Password Encryption Utility arrow to display the dialog box and encrypt the password.

See Also

[Chapter 4, “Managing Integration Gateways,” Encrypting Passwords, page 44](#)

CHAPTER 10

Using the Integration Point Wizard

This chapter discusses the Integration Point Wizard and describes how to:

- Create integrations using the Integration Point Wizard.
- Access relationships created by the Integration Point Wizard.

Understanding the Integration Point Wizard

The Integration Point Wizard guides you through the process of setting up point-to-point integrations, including specifying sending and receiving nodes, transactions, transformations, request messages, response messages, and so on. After you make your selections, the Integration Point Wizard creates the point-to-point integration, including any required relationships.

The Integration Point Wizard is for one-time use; after you create and save an integration in the wizard, you must use the appropriate PeopleSoft Application Designer and PeopleSoft Pure Internet Architecture pages to make any subsequent modifications to the integration point.

The Integration Point Wizard assists you in designing integration points. Additional tasks are required to run integrations you design using the Integration Point Wizard. For example you must configure the integration gateway, the pub/sub system must be booted (for asynchronous messaging), nodes must be active, message channels must be active, request and response messages must be active, sending and receiving PeopleCode must be written and associated with request and response messages, and so on.

The Integration Point Wizard is located in the Integration Point Wizard component (PT_IBSETUPWIZARD). To access the wizard, select PeopleTools, Integration Broker, Integration Point Wizard.

Prerequisites for Using the Integration Point Wizard

To create a point-to-point integration using the Integration Point Wizard, you must have the integration gateway set up and configured. In addition, you must have the following records defined:

- Sending node.
- Receiving node.
- Request message.
- Response message. (Synchronous messages only.)
- Request transformation. (If you are using transformations.)
- Response transformation. (If you are using transformations.)

- Asynchronous message reply. (For asynchronous-to-synchronous transactions only.)

Common Elements Used in This Chapter



This graphic shows the number of pages in the Integration Point Wizard. The number of current page displays in a yellow circle. Page numbers blacked out indicate those that you have completed and those that are not required for the transaction type you selected.

You can pause the cursor over a page number to display text that summarized the function of the page.

You can also click a page that you have already completed to go back to view and modify the contents of the page.

Click this button to view the next page in the wizard.

Click this button to view the previous page in the wizard

Click this button to cancel and exist a page in the wizard.

Click this button to save the integration and return to page one of the wizard.

Creating Integrations Using the Integration Point Wizard

This section describes creating integrations using the Integration Point Wizard and how to use the pages of the wizard to:

- Specify nodes and transaction types.
- Specify sender message information.
- Specify transformation information.
- Specify receiver message information.
- Specify asynchronous reply message and transformation information.
- View and save integration points.

Understanding Creating Integrations Using the Integration Point Wizard

This section discusses creating integrations using the Integration Point Wizard.

Integration Point Wizard Transaction Types

You can use several transaction types in conjunction with the Integration Point Wizard to create inbound, outbound, local-to-local, and hub integration points. The Integration Point Wizard supports the following transaction types

- Asynchronous** An asynchronous integration is one in which the sending system sends an outbound message to another system and does not require a response or any additional processing from the receiving system before processing on the sending system continues.
- Synchronous** A synchronous integration is one in which the sending system sends an outbound message to another system and requires a response or additional processing from the receiving system before processing on the sending system continues.
- Asynchronous-to-synchronous** An asynchronous-to-synchronous integration is one in which an asynchronous system sends a message to a synchronous system.
Since in this situation the receiving synchronous system sends a response back to the sending asynchronous system, the sending asynchronous system must be set up to receive a response. If the sending system is not set up to receive a response, processing will stop on the synchronous system.

Understanding Specifying Nodes

When you specify nodes in the Integration Point Wizard, you determine whether the integration is an outbound, inbound, local-to-local, or hub integration.

The following table describes how to define nodes based on the type of integration you want to create.

Sender Node	Receiver Node	Integration
Local node A.	Remote node B.	Outbound.
Remote node B.	Local node A.	Inbound.
Local node A.	Local node A.	Local-to-local.
Remote node B.	Remote node C.	Hub.

External Names

On several pages in the Integration Point Wizard, an optional External Name field is provided for both request and response messages.

The external name for a request message or response message is the message name that the external system is expecting. Use this field in situations where the receiving system does not understand the PeopleSoft message name. The name you enter is the message name sent to the external system.

An external name is used only when a response message is sent back to the receiver node. As an example, if a PeopleSoft systems receives a request and sends back a response message, it will use the external message name, if you have supplied one.

Transactions

An integration point may be comprised of several transactions. When using the Integration Point Wizard, you cannot specify a transaction modifier that already exists in the system.

Transformations

If an integration you create using the Integration Point Wizard requires transformation, all you need do is specify the transformation (and any transaction and connector overrides) and the wizard automatically creates the relationship.

After you create an integration that has a transformation, you can access the relationship in the Relationships component in the PeopleSoft Pure Internet Architecture.

Specifying Nodes and Transaction Types

The first step in creating an integration using the Integration Point Wizard is to access the Specifying Nodes and Transaction Types page to specify the sending node, the receiving node and the transaction type for the integration.

To access the page, select PeopleTools, Integration Broker, Integration Point Wizard.

Integration Point Wizard

1 2 3 4 5

< Previous Next > Finish Cancel

Specify Nodes and Transaction Type

*Sender Node: QE_LOCAL

*Receiver Node: QE_REMOTE

Transaction Type

☒ Asynchronous

☐ Synchronous

☐ Async To Sync

Specify Nodes and Transaction Type page

To specify nodes and the transaction type:

1. In the Sender Node field, select the sending node.
2. In the Receiver Node field, select the receiving node.
3. In the Transaction Type section, select the transaction type for the integration. The options are:
 - Asynchronous
 - Synchronous
 - Async To Sync (Asynchronous to Synchronous)

- Click the Next button to proceed to the next page in the wizard, the Specify Sender Message Information page.

Specifying Sender Message Information

Use the Specifying Sender Message Information page to define the request message and the request message version for the sending system. You may also specify a separate message name that the external system expects to receive.

If you are working with asynchronous and asynchronous-to-synchronous transaction types, you use this page to specify information for the request message only. The request message you specify is the message name before any transformations.

Integration Point Wizard

1 2 3 4 5

< Previous Next > Finish Cancel

Specify Sender Message Information

Sender Node

Request Message

*Name: QE_ASYNC_REQ_MSG 🔍

*Version: VERSION_1 🔍

External Name:

Specify Sender Message Information page for asynchronous and asynchronous-to-synchronous transaction types

If you are working with a synchronous transaction type, you specify information for the both the request message and the response message. The response message that you specify is the response message that the sending system ultimately consumes, *after* any transformations.

Integration Point Wizard

1 2 3 4 5

< Previous Next > Finish Cancel

Specify Sender Message Information

Sender Node

Request Message

*Name: QE_SYNC_REQ_MSG 🔍

*Version: VERSION_1 🔍

External Name:

Response Message

*Name: QE_SYNC_RES_MSG 🔍

*Version: VERSION_1 🔍

External Name:

Specify Sender Message Information page for a synchronous transaction type

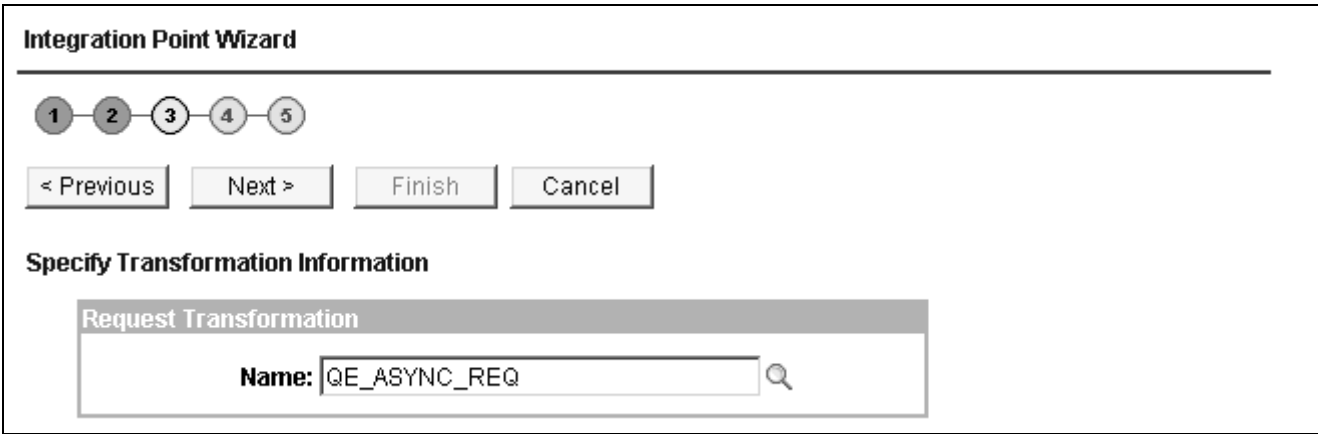
To specify sender message information:

1. In the Request Message section:
 - a. In the Name field, select a request message.
 - b. In the Version field, select the version of the request message to use.
 - c. (Optional) In the External Name field, enter the request message name that the external system is expecting.
2. If working with a synchronous transaction, in the Response Message section:
 - a. In the Name field, select a response message.
 - b. In the Version field, select the version of the response message to use.
 - c. (Optional) In the External Name field, enter the response message name that the external system is expecting.
3. Click the Next button to proceed to the next page, Specify Transformation Information page.

Specifying Transformation Information

Completing this page in the wizard is required only if the request message or the response message requires data transformation.

If you are working with an asynchronous transaction type, the page provides a field for the request transformation only.



Integration Point Wizard

1 2 3 4 5

< Previous Next > Finish Cancel

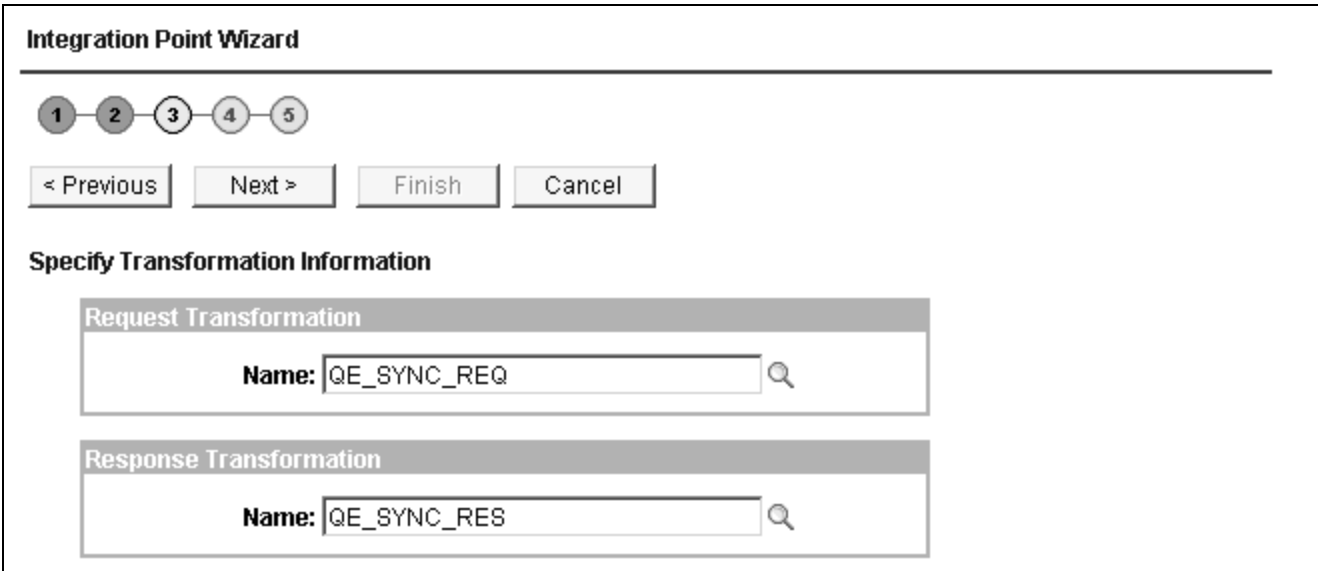
Specify Transformation Information

Request Transformation

Name: QE_ASYNC_REQ

Specify Transformation Information page for an asynchronous transaction

If you are working with a synchronous transaction type, the page provides fields for a request transformation and a response transformation.



Integration Point Wizard

1 2 3 4 5

< Previous Next > Finish Cancel

Specify Transformation Information

Request Transformation

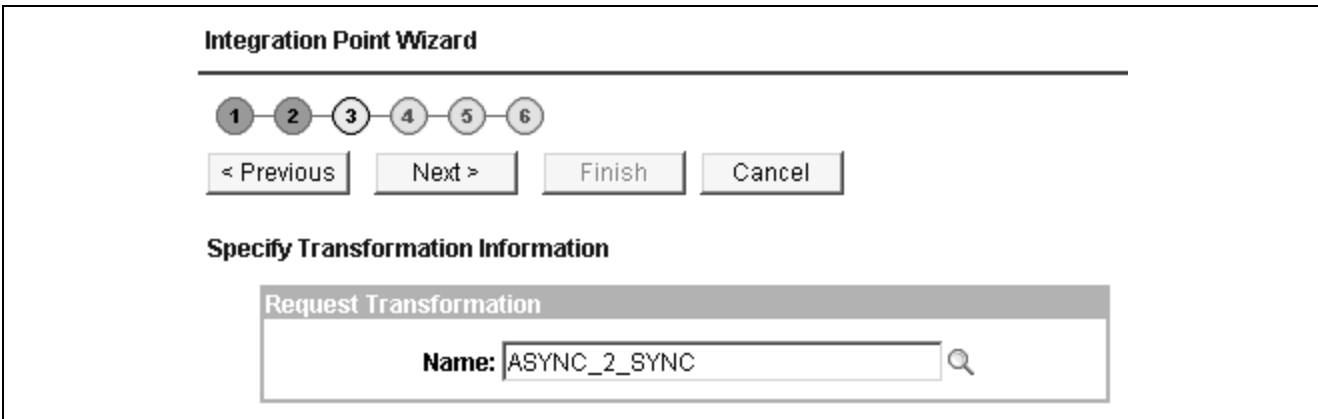
Name: QE_SYNC_REQ

Response Transformation

Name: QE_SYNC_RES

Specify Transformation Information page for a synchronous transaction

When working with an asynchronous-to-synchronous transaction, you specify a request transformation that transforms the asynchronous message into the shape of a synchronous message prior to sending the message to the receiving system.



Integration Point Wizard

1 2 3 4 5 6

< Previous Next > Finish Cancel

Specify Transformation Information

Request Transformation

Name: ASYNC_2_SYNC

Specify Transformation Information page for an asynchronous-to-synchronous transaction

In this situation, the request transformation for an asynchronous-to-synchronous transaction allows the message to be published asynchronously, whereby the sending system continues processing without receiving a response from the receiving system. However, the message is processed synchronously and the sending system returns a response message to the sending system.

To specify transformation information:

1. In the Request Transformation section, in the Name field, select a transformation for the request message.
2. If applicable, in the Response Transformation section, in the Name field, select a transformation for the response message.
3. Click the Next button to proceed to the next page, Specify Receiver Message Information page.

Specifying Receiver Message Information

In many situations, after a transformation has been applied to a message, the message name or message version changes.

Use the Specify Receiver Message Information page to specify post-transformation request message information for the receiving system. You may also specify external message names that the external system expects to receive.

The Specify Receiver Message Information page also enables you to override the default routing type and connector properties at the transaction level.

Note. If you did not specify any transformations on the previous Specify Transformation Information page, use this page to review the transaction information before proceeding to the next page.

If you are working with an asynchronous transaction, the Specify Receiver Message Information page displays and you can enter post-transformation request message information.

The screenshot shows the 'Integration Point Wizard' interface. At the top, there is a progress bar with five steps: 1, 2, 3, 4 (selected), and 5. Below the progress bar are four buttons: '< Previous', 'Next >', 'Finish', and 'Cancel'. The main title of the page is 'Specify Receiver Message Information'. Below this title is a form titled 'Receiver Node'. Inside the 'Receiver Node' form, there is a section titled 'Request Message'. This section contains three input fields: '*Name:' with the value 'ASYNC_REMOTE_REQ_MSG', '*Version:' with the value 'VERSION_1', and 'External Name:' which is empty. Each of the first two fields has a magnifying glass icon to its right. At the bottom right of the form, there is a link that says 'Go To Advanced Page'.

Specify Receiver Message Information page for an asynchronous transaction

If you are working with synchronous or asynchronous-to-synchronous transaction types, this page displays fields for you to specify information for the request message delivered to the receiving system, including any transformations, as shown in the following graphic. In the case of an asynchronous-to-synchronous transaction, the Name field in the Request Message section reflects the name of the asynchronous request message that was transformed into a synchronous message.

The Name field in the Response Message section is the name of the response message the receiving system sends back.

Integration Point Wizard

1 2 3 4 5

< Previous Next > Finish Cancel

Specify Receiver Message Information

Receiver Node

Request Message

*Name: SYNC_REMOTE_REQ_MSG

*Version: VERSION_1

External Name:

Response Message

*Name: SYNC_REMOTE_RES_MSG

*Version: VERSION_1

External Name:

[Go To Advanced Page](#)

Specify Receiver Message Information page for synchronous and asynchronous-to-synchronous transactions

If you click the Go To Advanced Page link, the Specify Transaction Properties page displays where you can override the routing type, logging level (synchronous transactions only), and connector.

Note. The Go to Advanced Page link does not display if you are working with an inbound asynchronous transaction.

Specifying Receiver Message Information

To specify receiver message information:

1. In the Name field, select the name of the transformed request message for the receiving node.
2. In the Version field, select the message version
3. (Optional) In the External Name field, enter the request message name that the external system is expecting.
4. You have two options:

- To override the default transaction properties and specify connector overrides, click the Go To Advanced Page link. The Specify Transaction Properties page displays.
- If you do not want to override transaction properties or connectors, click the Next button to proceed to the next page in the wizard.

If you are working with an asynchronous or synchronous transaction, the next page in the wizard is the Summary page where you can view details of the completed integration and save it.

If you are working with an asynchronous-to-synchronous transaction, the next page in the wizard is the Specify Asynchronous Reply Message and Transformation page.

Specifying Transaction Properties

Overriding transaction properties is optional.

The following graphic shows the Specify Transaction Properties page that displays when working with an asynchronous transaction. Use this page to override the transaction routing type, override the connector, and override connector properties. This page also features access to the Password Encryption Utility, so you can encrypt any password-protected connector properties that you choose to override.

The image shows a dialog box titled "Specify Transaction Properties". It contains a section labeled "Request Properties" with a dropdown menu for "*Routing Type:" set to "Implicit". Below this is a checkbox labeled "Override Connector". Underneath the checkbox is a section titled "Transaction Connectors" which contains two text input fields: "*Gateway ID:" and "*Connector ID:". At the bottom of the dialog is a button labeled "Password Encryption Utility" with a right-pointing arrow. In the bottom right corner of the dialog are "Done" and "Cancel" buttons.

Specify Transaction Properties page for an asynchronous transaction

When you select the Override Connector box, the Transaction Connector section of the page becomes enabled and you can work with connectors and connector properties. You override request connector properties usually when integrating with third-party systems.

To access the encryption utility, click the Password Encryption Utility arrow.

Specify Transaction Properties

Request Properties

*Routing Type: Implicit

☒ **Override Connector**

Transaction Connectors

*Gateway ID:

*Connector ID:

Properties

View All First 1 of 1 Last

*Property ID	*Property Name	Value
1		

▶ Password Encryption Utility

Specify Transaction Properties page with the Transaction Connectors section enabled

When working with a synchronous transaction, the Specify Transaction Properties page also displays a Synchronous Logging section where you can override the logging setting.

Specify Transaction Properties

Synchronous Logging

*Logging: No Logging

Request Properties

*Routing Type: Implicit

☐ **Override Connector**

Transaction Connectors

*Gateway ID:

*Connector ID:

▶ Password Encryption Utility

Specify Transaction Properties page for synchronous and asynchronous-to-synchronous transactions

To specify transaction properties:

1. (Optional) To override the synchronous logging level, from the Logging drop-down list, select an option.
2. (Optional) To override the request transaction routing type, in the Request Properties box, from the Routing drop-down list, select the appropriate routing type.
3. (Optional) To override connector properties, select the Override Connector box.

link in the Request Properties section or in the Response properties section of the page.

If you do not want to override connector properties, click the Done button to apply and transaction properties changes you made. The wizard returns you to the Specify Receiver Message Information page.

Click the Next button to proceed to the next page in the wizard, the Summary page, to view details of the integration and save it.

- a. In the Gateway ID field, select the gateway of the sending system.
- b. In the Connector ID field, select the desired target connector from the list.
When you select the connector, its required properties and their current values display in the Properties section.
- c. In the Properties section, you can change the values of existing properties and add properties.
- d. Click the Done button to apply the changes and return to the Specify Receiver Message Information page.
- e. Click the Next button to proceed to the next page in the wizard.

To exit this page without applying changes, click the Cancel button.

To delete a connector override after you have clicked the Done button, clear the Override Connector box.

See [Chapter 9, “Configuring Nodes and Transactions,” Creating Node Definitions, page 196.](#)

See [Chapter 9, “Configuring Nodes and Transactions,” Editing Transaction Message Details, page 211.](#)

Specifying Asynchronous Reply Message and Transformation

The Specify Async Reply Message and Transformation page of the Integration Point Wizard displays only when you are working with an asynchronous-to-synchronous transaction.

Integration Point Wizard

1 2 3 4 5 6

< Previous Next > Finish Cancel

Specify Async Reply Message and Transformation

Async Reply Message

*Name: QE_ASYNC_REPLY_MSG 🔍

*Version: VERSION_1 🔍

Transformation: SYNC_2_ASYNC 🔍

Specify Async Reply Message and Transformation page

This page enables you to specify the name of the asynchronous message that will be run locally with the content data received from the target system. A transformation is invoked to transform the receiving data into the shape of an asynchronous reply message.

To specify asynchronous reply message and transformation information:

1. In the Name field, select the asynchronous reply message name.
2. In the Version field, select the version number for the message.
3. (Optional) In the Transformation field, select the transformation for the reply message.

4. Click the Next button to proceed to the next page in the wizard, the Summary page.

Viewing and Saving Integrations

The Summary page of the Integration Point Wizard enables you to view all the details of an integration and save it.

As you view an integration point, notice that the numbered boxes correspond to the numbered pages in the wizard.

Warning! Take time to carefully review integrations before you click the Finish button.

After you save an integration created using the Integration Point Wizard, you cannot use the wizard to make subsequent changes to the integration. Instead, you must use the appropriate PeopleSoft Application Designer pages or PeopleSoft Pure Internet Architecture pages to make any changes or modifications. Use the Previous button or the page number links at the top of the page to go back to pages in the wizard to make changes.

After you have reviewed an integration and are satisfied with the results, click the Finish button to save it. You then have the option of opening the integration in the Integration Point Viewer or creating another integration point in the Integration Point Wizard.

Viewing Asynchronous Integration Points

The following graphic shows summary information for an asynchronous transaction, based on the input values shown in the previous sections of this chapter.

Integration Point Wizard

1 2 3 4 5

< Previous Next > Finish Cancel

Summary

Transaction Type: Asynchronous

Sender Node

Name: QE_LOCAL

Receiver Node

Name: QE_REMOTE

Request Routing Type: Implicit

1. Request Message

Name: ASYNC_REQ_MSG

Version: VERSION_1

External Name:

Transformations

Request: QE_ASYNC_REQ

2. Request Message

Name: ASYNC_REMOTE_REQ_MSG

Version: VERSION_1

External Name:

Summary page for an asynchronous integration

The page shows that the sending node is *QE_LOCAL* and the receiving node is *QE_REMOTE*. Since the sending node is the local node and the receiving node is a remote node, this is an outbound asynchronous integration point.

The page shows the following about the outbound request:

1. The box labelled 1. Request Message shows the original request message is *QE_ASYNC_REQ_MSG*.
2. The Transformation box shows that the transformation *QE_ASYNC_REQ* was applied to the *ASYNC_REQ_MSG* message.
3. The box labelled 2. Request Message shows that the transformation *QE_ASYNC_REQ* has transformed the message *ASYNC_REQ_MSG* to *ASYNC_REMOTE_REQ_MSG*.
4. As a result, the message *ASYNC_REMOTE_REQ_MSG* is the message that is sent to the receiving node, *QE_REMOTE*.

Viewing Synchronous Integration Points

The following graphic shows the Summary page a synchronous transaction, based on the input values shown in previous sections of this chapter.

Integration Point Wizard

1 2 3 4 5

< Previous Next > Finish Cancel

Summary

Transaction Type: Synchronous **Logging:** Header Only

Sender Node

Name: QE_LOCAL

Receiver Node

Name: QE_REMOTE

Request Routing Type: Implicit

1. Request Message	4. Response Message
Name: QE_SYNC_REQ_MSG	Name: QE_SYNC_RES_MSG
Version: VERSION_1	Version: VERSION_1
External Name:	External Name:

Transformations

Request: QE_SYNC_REQ	Response: QE_SYNC_RES
2. Request Message	3. Response Message
Name: SYNC_REMOTE_REQ_MSG	Name: SYNC_REMOTE_RES_MSG
Version: VERSION_1	Version: VERSION_1
External Name:	External Name:

Summary page for a synchronous integration point

For synchronous integration points, view the information in a u-shape fashion, starting at the top left-side of the page, reading downward to the bottom, rightward across the bottom of the page, and then upward on the right-side of the page.

The page shows that the sending node is *QE_LOCAL* and the receiving node is *QE_REMOTE*. Since the sending node is the local node and the receiving node is a remote node, this is an outbound synchronous integration point.

The page shows the following about the outbound request:

1. The box labelled 1. Request Message shows the original request message is *QE_SYNC_REQ_MSG*.

2. Moving down the page, the Transformation box shows that the request transformation *QE_SYNC_REQ* was applied to the message *QE_SYNC_REQ_MSG*.
3. Continuing down the page, the box labelled 2. Request Message shows that the transformation transformed the message into the message *SYNC_REMOTE_REQ_MSG*. As a result, the message *SYNC_REMOTE_REQ_MSG* is the message that is sent to the receiving node, *QE_REMOTE*.

The page shows the following about the returned response:

1. The box labelled 3. Response Message shows the initial message response received from the receiving system is *SYNC_REMOTE_RES_MSG*.
2. Moving up the page directly above that box, the Transformation box shows that a response transformation *QE_SYNC_RES* was applied to the message.
3. Continuing up the page, the box labelled 4. Response Message shows that the transformation transformed the message into the message *QE_SYNC_RES_MSG*. As a result, the message *QE_SYNC_RES_MSG* is the response message that is sent to the sending node, *QE_LOCAL*.

Viewing Asynchronous-to-Synchronous Integration Points Created

The following graphic shows summary information for an asynchronous-to-synchronous transaction, based on the input values shown in the previous sections of this chapter.

Integration Point Wizard

1 2 3 4 5 6

< Previous Next > Finish Cancel

Summary

Transaction Type: Async To Sync **Logging:** Header Only

Sender Node

Name: QE_LOCAL

Receiver Node

Name: QE_REMOTE

Request Routing Type: Implicit

1. Async Request Message Name: QE_REQ_ASYNC_MSG Version: VERSION_1 External Name:	4. Async Response Message Name: QE_ASYNC_REPLY_MSG Version: VERSION_1
Transformations Request: ASYNC_2_SYNC Response: SYNC_2_ASYNC	
2. Sync Request Message Name: SYNC_REMOTE_REQ_MSG Version: VERSION_1 External Name:	3. Sync Response Message Name: SYNC_REMOTE_RES_MSG Version: VERSION_1 External Name:

Summary page for an asynchronous-to-synchronous integration point

For asynchronous-to-synchronous integration points, view the information in a u-shape fashion, starting at the top left-side of the page, reading downward to the bottom, rightward across the bottom of the page, and then upward on the right-side of the page.

The page shows that the sending node is *QE_LOCAL* and the receiving node is *QE_REMOTE*. Since the sending node is the local node and the receiving node is a remote node, this is an outbound asynchronous-to-synchronous integration point.

The page shows the following about the outbound request message:

1. The box labelled 1. Request Message shows the original request message is *QE_REQ_ASYNC_MSG*.
2. The Transformation box shows that the transformation *ASYNC_2_SYNC* was applied to the message *QE_REQ_ASYNC_MSG*.
3. The box labelled 2. Request Message shows that the transformation *ASYNC_2_SYNC* transformed the message *QE_REQ_ASYNC_MSG* to the shape of a synchronous message with the name *SYNC_REMOTE_REQ_MESG*.

As a result, the message *SYNC_REMOTE_REQ_MESG* is the message that is sent to the receiving node, *QE_REMOTE*.

The page shows the following about the response message:

1. The box labelled 3. Response Message shows the initial message response received from the receiving system is *SYNC_REMOTE_RES_MESG*.
2. Moving up the page directly above that box, the Transformation box shows that a response transformation *SYNC_2_ASYNC* was applied to the message *SYNC_REMOTE_RES_MESG*
3. Continuing up the page, the box labelled 4. Response Message shows that the transformation *SYNC_2_ASYNC* transformed the *SYNC_REMOTE_RES_MESG* message into the message *QE_ASYNC_REPLY_MSG*.

As a result, the message *QE_ASYNC_REPLY_MSG* is the reply that is sent to the sending node, *QE_LOCAL*.

Accessing Relationships Created by the Integration Point Wizard

After you create an integration that has a transformation, you can access the relationship in the Relationships component in the PeopleSoft Pure Internet Architecture.

To access the component, select PeopleTools, Integration Broker, Relationships. When you click the Search button, relationships created with the wizard are named using the following convention: WIZARD <date created in yyyy/mm/dd format> <unique identifier>.

Search Results		
View All		
First 1-9 of 9 Last		
Relationship ID	Node Name	Node Name
QE_ASYNC_RELATIONSHIP	QE_IBTGT	QE_IBTGT
QE_CUSTOMER	QE_SOAPSWITCH	QE_SOAPSWITCH
QE_IB_ONQELOCAL_IBTGT	QE_IBTGT	QE_IBTGT
QE_IB_ONQELOCAL_LOCAL	QE_LOCAL	QE_LOCAL
QE_SYNC_RELATIONSHIP	QE_IBTGT	QE_IBTGT
WIZARD2004 03 03 13 37 38 0000	QE_UNDERDOG	QE_YAHOOSQ
WIZARD2004 03 08 13 40 04 0000	QE_REMOTE	QE_REMOTE
WIZARD2004 03 08 13 50 23 0000	QE_REMOTE	QE_REMOTE
WIZARD2004 03 08 13 57 21 0000	QE_REMOTE	QE_REMOTE

Relationship search results

The previous graphic shows a relationship search that returned nine results. The last four relationships in the list were created with the Integration Point Wizard.

If you select, say, the last relationship in the list, you can use the Relationship component pages to view and manipulate that relationship as you would with any other relationship.

Relationships **Transaction Modifiers**

Relationship ID WIZARD2004_03_08_13_57_21_0000

Details

***Description** Created by Integration Point W

***Relationship Status** Active

Node

***Node Name** QE_REMOTE [Properties](#) [Contact](#)

Node

***Node Name** QE_REMOTE [Properties](#) [Contact](#)

Comment

Nodes page in the Relationship component

Notice that the pages are populated based on the data and values you selected and entered in the Integration Point Wizard.

Relationships **Transaction Modifiers**

Relationship ID WIZARD2004_03_08_13_57_21_0000

Node Name QE_REMOTE

Node Name QE_REMOTE

Transaction Modifiers [Customize](#) | [Find](#) | [First](#) | [1 of 1](#) | [Last](#)

<u>Initial Node</u>	<u>Transaction Type</u>	<u>Request Message Name</u>	<u>Effective Date</u>	<u>Status</u>
1 Edit QE_REMOTE	OA	QE_REQ_ASYNC_MSG	03/08/2004	Active

Relationships-Trans Modifiers page

If you need to modify the relationship, click the Edit link for the transaction, and make modifications as you would for any relationship.

Relationships

Transaction Modifiers

Relationship ID

WIZARD2004_03_08_13_57_21_0000

Node Name

QE_REMOTE

Node Name

QE_REMOTE

Effective Date

03/08/2004

*Status

Active

Sequence number

1

Initial

Node

QE_REMOTE

Transaction Type

OA

Request Message Name

QE_REQ_ASYNC_MSG

Version

VERSION_1

Result

Node

QE_REMOTE

*Transaction Type

OS

Request Message Name

SYNC_REMOTE_REQ_MSG

Version

VERSION_1

Transformations

Request

ASYNC_2_SYNC

Response

SYNC_2_ASYNC

Async Reply Message

QE_ASYNC_REPLY_MSG

Async Reply Message Version

VERSION_1

[Return to Transaction List](#)

Viewing the details of the relationship

See Also

[Chapter 9, “Configuring Nodes and Transactions,” Editing Transaction Details, page 209](#)

[Chapter 13, “Administering Relationships,” Managing Transaction Modifiers, page 281](#)

CHAPTER 11

Using the Integration Point Viewer

This chapter describes the Integration Point Viewer and discusses how to:

- Access integrations.
- View integration information.
- View inactive objects.
- Activate inactive objects.

Understanding the Integration Point Viewer

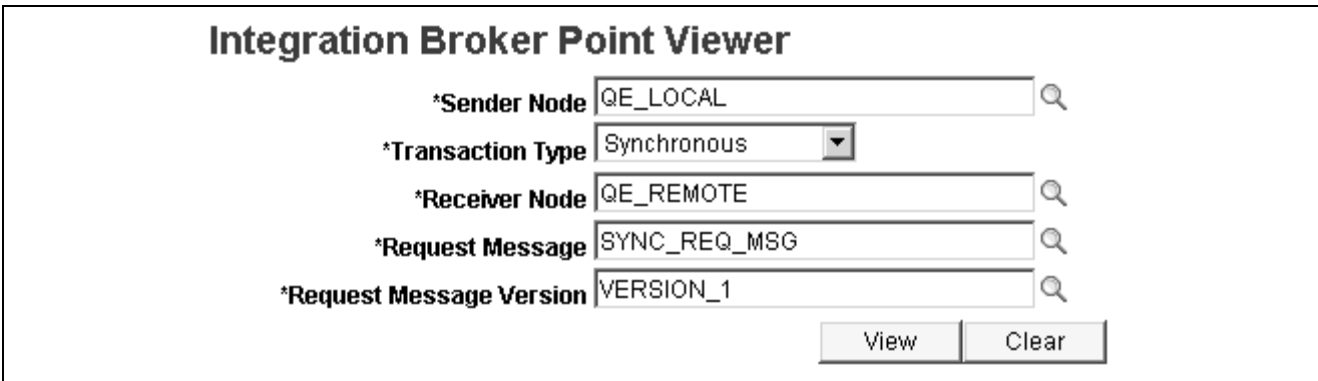
The Integration Point Viewer is a read-only tool located in the Integration Point Viewer component (PT_IBPOINTVIEWER) that enables you to display a point-to-point view of an integration to validate its structure and confirm that the objects that comprise it are active.

Depending on the how the integration is set up and the transaction type, the Integration Point Viewer provides direct links to the node definition, the transaction and the relationship where you can view additional information and make modifications.

Accessing Integrations

To access an integration using the Integration Point Viewer, select PeopleTools, Integration Broker, Integration Point Viewer.

Initially, the search section of the Integration Broker Point Viewer page displays where you enter criteria for the integration to view.



The screenshot shows the 'Integration Broker Point Viewer' search interface. It contains five input fields with labels: '*Sender Node' (containing 'QE_LOCAL'), '*Transaction Type' (a dropdown menu showing 'Synchronous'), '*Receiver Node' (containing 'QE_REMOTE'), '*Request Message' (containing 'SYNC_REQ_MSG'), and '*Request Message Version' (containing 'VERSION_1'). Each text input field has a magnifying glass icon to its right. At the bottom right of the form are two buttons: 'View' and 'Clear'.

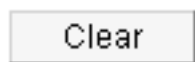
Search section of the Integration Point Viewer page

After you enter the search criteria and click the View button, summary information for the transaction displays.

Common Elements Used to Access Integration Points



Click the View button to view an integration based on the criteria entered in the Integration Point Viewer.



Click the Clear button to clear the integration search criteria or the summary information for the displayed integration.

Accessing Integration Points

To access an integration point:

1. Select PeopleTools, Integration Broker, Integration Point Viewer.
2. In the Sender Node field, select the sending node.
3. From the Transaction Type drop-down list, select the type of transaction. The valid values are:
 - Asynchronous
 - Async to Sync (Asynchronous-to-Synchronous)
 - Synchronous
4. In the Receiver Node field, select the receiving node.
5. In the Request Message field, select the name of the request message.
Integration Point Viewer displays a list of messages from which you can select, based on the node and transaction type selected.
6. In the Request Message Version field, select the version of the request message.
7. Click the View button to access and view the integration point.

Viewing Integration Information

After you enter the details about an integration and click the View button, the Integration Point Viewer displays summary information about the integration.

When viewing information for asynchronous integration points, view the information from top to bottom. When viewing information for synchronous and asynchronous-to-synchronous integration points, view the information in a u-shape fashion, starting at the top left-side of the page, reading downward to the bottom, rightward across the bottom of the page, and then upward on the right-side of the page.

Common Elements Used to View Integration Point Information

Go to Inactive Objects Page This link displays when one or more objects associated with an integration are inactive.

Click the link to open the Inactive Objects page to view and activate any inactive objects.

Note. If any object associated with an integration is inactive, the status of the integration is *Inactive*.

Go to Node Transaction

Click this link to open the Transaction Detail page for the node. The Transaction Detail page opens in a separate window, where you can view and change information about the transaction, such as its effective date, status, routing type, connector overrides, and so on.

When you are done working with the information, save any changes and click the Close button at the top right corner of the window.

Go to Transaction Modifier

Click this link to open the Trans Modifiers page for the relationship. The Trans Modifiers page opens in a separate window, where you can view and change information about the relationship.

When you are done working with the information, save any changes and click the Close button at the top right corner of the window.

Viewing Asynchronous Integration Point Information

This section discusses viewing integration point information for asynchronous integration points.

Integration Broker Point Viewer

*Sender Node 
 *Transaction Type
 *Receiver Node 
 *Request Message 
 *Request Message Version 

Status: Inactive [Go to Inactive Objects Page](#)

Sender Node

Name: QE_LOCAL

Receiver Node

Name: QE_REMOTE

Request Routing Type: Implicit

1. Request Message

Name: QE_ASYNC_REQ_MSG

Version: VERSION_1

External Name:

[Go to Node Transaction](#)

Transformations

Request: QE_ASYNC_REQ

[Go To Transaction Modifier](#)

2. Request Message

Name: ASYNC_REMOTE_REQ_MSG

Version: VERSION_1

External Name:

[Go to Node Transaction](#)

Integration Point Viewer page displaying information for an asynchronous integration point

The integration point information in the previous graphic shows and that *QE_LOCAL* is the sending node and the receiving node is *QE_REMOTE*. Since the sending node is the local node and the receiving node is a remote node, this is an outbound asynchronous integration point.

The page shows the following about the outbound request:

1. The box labelled 1. Request Message shows the original request message is *QE_ASYNC_REQ_MSG*.
2. The Transformation box shows that the transformation *QE_ASYNC_REQ* was applied to the message *QE_ASYNC_REQ_MSG*.
3. The box labelled 2. Request Message shows that the transformation *QE_ASYNC_REQ* has transformed the message *QE_ASYNC_REQ_MSG* to *ASYNC_REMOTE_REQ_MSG*.

As a result, the message *ASYNCRREMOTEREQMSG* is the message that is sent to the receiving node, *QEREMOTE*.

Viewing Synchronous Integration Point Information

This section discusses viewing integration point information for synchronous integration points.

Integration Broker Point Viewer

*Sender Node

QE_LOCAL

*Transaction Type

Synchronous

*Receiver Node

QE_REMOTE

*Request Message

QE_SYNC_REQ_MSG

*Request Message Version

VERSION_1

View

Clear

Status: Inactive

[Go to Inactive Objects Page](#)

Logging: No Logging

Sender Node

Name: QE_LOCAL

Receiver Node

Name: QE_REMOTE

Request Routing Type: Implicit

1. Request Message

Name: QE_SYNC_REQ_MSG

Version: VERSION_1

External Name:

4. Response Message

Name: QE_SYNC_RES_MSG

Version: VERSION_1

External Name:

[Go to Node Transaction](#)

Transformations

Request: QE_SYNC_REQ

Response: QE_SYNC_RES

[Go To Transaction Modifier](#)

2. Request Message

Name: SYNC_REMOTE_REQ_MSG

Version: VERSION_1

External Name:

3. Response Message

Name: SYNC_REMOTE_RES_MSG

Version: VERSION_1

External Name:

[Go to Node Transaction](#)

Integration Point Viewer page displaying information for a synchronous integration point

For synchronous integration points, view the information in a u-shape fashion, starting at the top left-side of the page, reading downward to the bottom, rightward across the bottom of the page, and then upward on the right-side of the page.

The page shows that the sending node is *QE_LOCAL* and the receiving node is *QE_REMOTE*. Since the sending node is the local node and the receiving node is a remote node, this is an outbound synchronous integration point.

The page shows the following about the outbound request:

1. The box labelled 1. Request Message shows the original request message is *QE_SYNC_REQ_MSG*.
2. Moving down the page, the Transformation box shows that the request transformation *QE_SYNC_REQ* was applied to the message *QE_SYNC_REQ_MSG*.
3. Continuing down the page, the box labelled 2. Request Message shows that the transformation transformed the message into the message *SYNC_REMOTE_REQ_MSG*.

As a result, the message *SYNC_REMOTE_REQ_MSG* is sent to the receiving node, *QE_REMOTE*.

The page shows the following about the returned response:

1. The box labelled 3. Response Message shows the initial message response received from the receiving system is *SYNC_REMOTE_RES_MSG*.
2. Moving up the page directly above that box, the Transformation box shows that a response transformation *QE_SYNC_RES* was applied to the message.
3. Continuing up the page, the box labelled 4. Response Message shows that the transformation transformed the message into the message *QE_SYNC_RES_MSG*.

As a result, the message *QE_SYNC_RES_MSG* is the response message that is sent to the sending node, *QE_LOCAL*.

Viewing Asynchronous-to-Synchronous Integration Point Information

This section discusses viewing integration point information for asynchronous-to-synchronous integration points.

Integration Broker Point Viewer

*Sender Node 
 *Transaction Type
 *Receiver Node 
 *Request Message 
 *Request Message Version 

Status: Inactive [Go to Inactive Objects Page](#) **Logging:** No Logging

Sender Node

Name: QE_LOCAL

Receiver Node

Name: QE_REMOTE

Request Routing Type: Implicit

1. Async Request Message

Name: QE_REQ_ASYNC_MSG

Version: VERSION_1

External Name:

[Go to Node Transaction](#)

4. Async Response Message

Name: QE_ASYNC_REPLY_MSG

Version: VERSION_1

[Go to Node Transaction](#)

Transformations

Request: ASYNC_2_SYNC **Response:** SYNC_2_ASYNC

[Go To Transaction Modifier](#)

2. Sync Request Message

Name: SYNC_REMOTE_REQ_MSG

Version: VERSION_1

External Name:

3. Sync Response Message

Name: SYNC_REMOTE_RES_MSG

Version: VERSION_1

External Name:

[Go to Node Transaction](#)

Integration Point Viewer page displaying information for an asynchronous-to-synchronous integration point

For asynchronous-to-synchronous integration points, view the information in a u-shape fashion, starting at the top left-side of the page, reading downward to the bottom, rightward across the bottom of the page, and then upward on the right-side of the page.

The page shows that the sending node is *QE_LOCAL* and the receiving node is *QE_REMOTE*. Since the sending node is the local node and the receiving node is a remote node, this is an outbound asynchronous-to-synchronous integration point.

The page shows the following about the outbound request message:

1. The box labelled 1. Request Message shows the original request message is *QE_REQ_ASYNC_MSG*.

2. The Transformation box shows that the transformation *ASYNC_2_SYNC* was applied to the message *QE_REQ_ASYNC_MSG*.
3. The box labelled 2. Request Message shows that the transformation *ASYNC_2_SYNC* transformed the message *QE_REQ_ASYNC_MSG* to the shape of a synchronous message with the name *SYNC_REMOTE_REQ_MESG*.

As a result, the message *SYNC_REMOTE_REQ_MESG* is the message that is sent to the receiving node, *QE_REMOTE*.

The page shows the following about the response message:

1. The box labelled 3. Response Message shows the initial message response received from the receiving system is *SYNC_REMOTE_RES_MSG*.
2. Moving up the page directly above that box, the Transformation box shows that a response transformation *SYNC_2_ASYNC* was applied to the message.
3. Continuing up the page, the box labelled 4. Response Message shows that the transformation transformed the message into the message *QE_ASYNC_REPLY_MSG*.

As a result, the message *QE_ASYNC_REPLY_MSG* is the reply message that is used locally on the sending node, *QE_LOCAL*.

Viewing Inactive Objects

When any objects associated with an integration are inactive, the View Inactive Objects link displays when you view the integration. When you click the View Inactive Objects link, the Inactive Objects page displays. The page lists all of the inactive objects for the integration.

Inactive Objects

Node		View All		First 1 of 1 Last	
Edit	Node Name				
1 Edit	QE_REMOTE				

Transactions					View All		First 1 of 1 Last	
Edit	Transaction Type	Request Message	Request Message Version	Effective Date				
1 Edit	OutSync	QE_SYNC_REQ_MSG	VERSION_1	11/09/2003				

Message		View All		First 1-2 of 2 Last	
Message Name	Active				
1 SYNC_REMOTE_REQ_MSG	☐	Activate selected messages			
2 QE_SYNC_REQ_MSG	☐				

Message Subscription		View All		First 1 of 1 Last	
Message Name	Subscription Name				
1 QE_SYNC_REQ_MSG	Test				

Message Channel		View All		First 1 of 1 Last	
Channel Name	Active				
1 IB_CHNL	☐	Activate selected channel			

Relationship		View All		First 1 of 1 Last	
Edit	Relationship ID				
1 Edit	WIZARD2003_11_09_12_50_09_0000				

Transaction Modifier						View All		First 1 of 1 Last	
Initial		Result							
Edit	Initial Node	Transaction Type	Request Message Name	Effective Date	Status				
1 Edit	QE_REMOTE	OS	QE_SYNC_REQ_MSG	11/09/2003	Inactive				

Refresh [Return to Viewer Page](#)

Inactive Objects page

Depending on how the integration is set up, the following objects related to an integration can appear on the Inactive Objects page:

- Nodes
- Transactions
- Messages
- Message channels
- Message subscriptions
- Transaction modifiers
- Relationships

Activating Inactive Objects

This section describes how to activate objects that display as inactive on the Inactive Objects page of the Integration Point Viewer.

Activating Nodes

To activate a node from the Inactive Objects page:

1. In the Node section, click the Edit link for the node to activate. A separate window opens and displays the Node Definitions page for the node.
2. Check the Active Node box.
3. Click the Save button.
4. Return to the Inactive Objects page and click the Refresh button.

Activating Transactions

To activate a transaction from the Inactive Objects page:

1. In the Transactions section, click the Edit link for the transaction to activate. A separate window opens and displays the Node Definition —Transaction Detail tab.
2. From the Status drop-down list, select Active.
3. Click the Save button.
4. Return to the Inactive Objects page and click the Refresh button.

Activating Messages

To activate messages from the Inactive Objects page:

1. In the Message section, check the Active box for each message to activate.
2. Click the Activate Selected Messages button.
3. Click the Refresh button.

Activating Message Subscriptions

To activate message subscription PeopleCode for a message:

1. In the Message Subscription section, check the Active box for each message subscription to activate.
2. Click the Activate Selected Subscription button.
3. Click the Refresh button.

Activating Message Channels

To activate a message channel from the Inactive Objects page:

1. In the Message Channel section, check the Active box for the message channel to activate.
2. Click the Activate Selected Channel button.

3. Click the Refresh button.

Activating Relationships

To activate a relationship from the Inactive Objects page:

1. In the Relationships section, click the Edit link for the relationship to activate. A separate window opens and displays the Relationships-Nodes tab.
2. From the Relationship Status drop-down list, select Active.
3. Click the Save button and close the page.
4. Return to the Inactive Objects page and click the Refresh button.

Activating Transaction Modifiers

To activate a transaction modifier, the relationship must first be active.

To activate transaction modifiers from the Inactive Objects page:

1. In the Transaction Modifier section, click the Edit link for the transaction modifier to activate. A separate window opens and displays the Relationships-Trans Modifier tab for the transaction.
2. In the Status drop-down list, select Active.
3. Click the Save button and close the page.
4. Return to the Inactive Objects page and click the Refresh button.

CHAPTER 12

Applying Filtering, Transformation and Translation

This chapter discusses how to:

- Develop transform programs.
- Filter messages and generate errors.
- Apply transformations.
- Perform data translation.

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.

To develop a transform program, you must know the initial structure and possibly the content of the message you're working with, 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 an Integration Broker node and a third-party application. Any participating node with 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 *transaction modifier*, which is associated with a *relationship definition*.

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.

See Also

[Chapter 5, "Understanding Supported Message Structures," page 57](#)

[Chapter 13, "Administering Relationships," page 273](#)

Enterprise PeopleTools 8.45 PeopleBook: Global Technology, "Selecting and Configuring Character Sets and Language Input and Output"

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 Integration Broker transformations to accommodate the third-party application.

Note. Third parties can submit messages to 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.

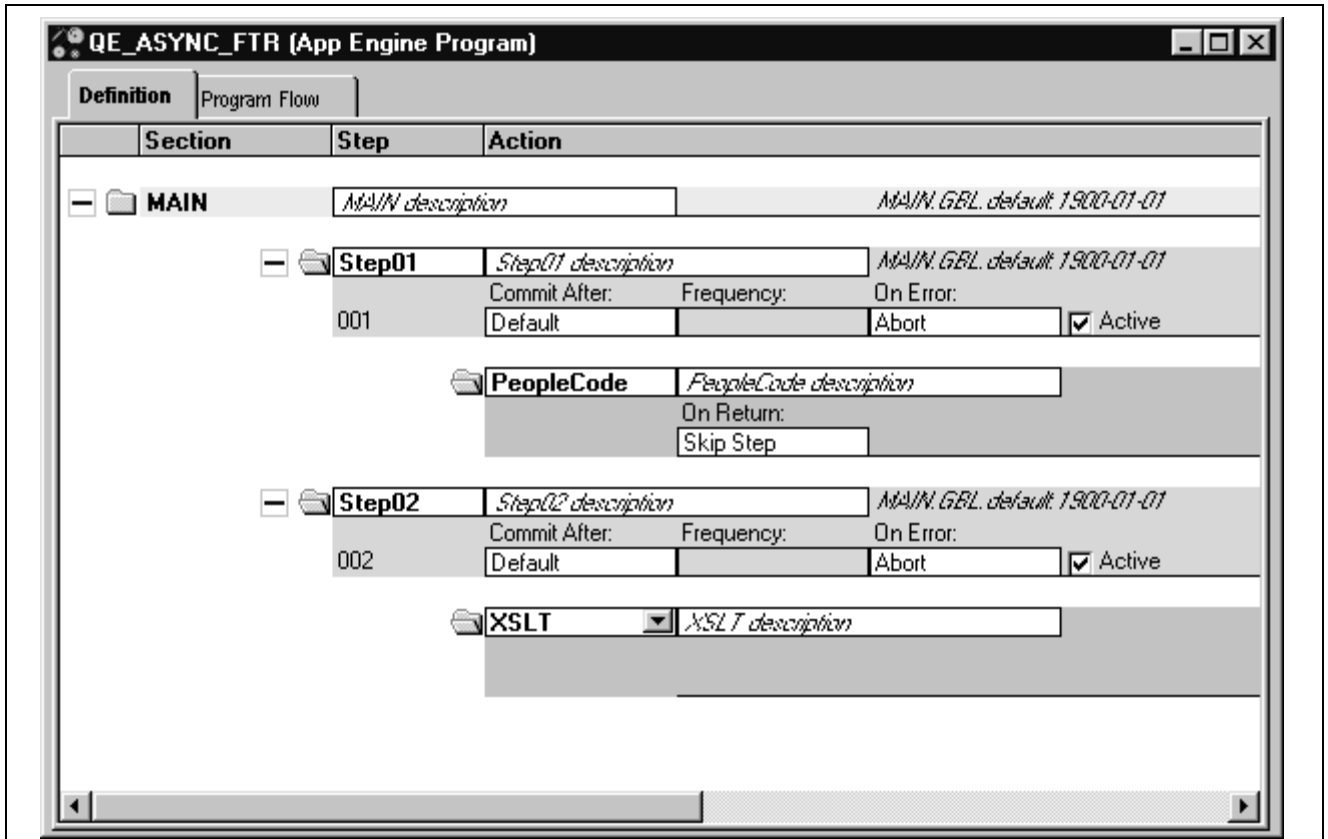
Developing Transform Programs

This section discusses how to:

- Define a transform program.
- Work with transform programs.
- Access message data.
- Make working data available globally.
- Preserve record and field aliases.

Defining Transform Programs

In PeopleSoft Application Designer, select File, New, App Engine Program to create a new program.



Application Engine transform program definition

To define a transform program:

1. Open the program properties and select the Advanced tab.
2. Select a program type of *Transform Only*, click OK and save the program.
3. Insert sections, steps and actions as needed.

Construct your program as you would any other Application Engine program, using actions of type *XSLT* and *PeopleCode*.

See Also

Enterprise PeopleTools 8.45 PeopleBook: PeopleSoft Application Engine, “Understanding PeopleSoft Application Engine”

Working with Transform Programs

PeopleSoft Integration Broker supports the use of Extensible Stylesheet Language Transformation (XSLT) code and PeopleCode, implemented in your transform program as actions of type *XSLT* or *PeopleCode*, respectively. Following are some points to keep in mind:

- 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.
- Transformations on outbound synchronous requests executed from within Application Engine programs (in batch) that contain one or more XSLT steps are not supported on the OS/390 and z/OS platforms, since Xalan is not available on these platforms. Xalan is an XSLT processor that transforms XML documents into HTML, text, or other XML document types.

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.

Deciding Which Language to Use

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.

However, you can't use XSLT to filter messages based on content, so filtering must be implemented in PeopleCode.

You can use both XSLT and PeopleCode steps in a single transform program.

Each XSLT program must be enclosed in the following 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.

Invoking Transform Programs

Your transform program is invoked by PeopleSoft Integration Broker if you specify its name in the Transformations group Request or Response field when you edit the details of a transaction modifier.

Renaming or Deleting Transform Programs

To invoke an Application Engine transform program, you specify it as part of a transaction modifier. If you subsequently rename or delete that transform program, it still appears in the transaction modifier definition, so transactions 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 transaction modifiers that apply it.
- If you delete a transform program, make sure you select a different program (or no program) in any transaction modifiers 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

Chapter 13, “Administering Relationships,” Configuring Relationships, page 277

Chapter 13, “Administering Relationships,” Managing Transaction Modifiers, page 281

Enterprise PeopleTools 8.45 PeopleBook: PeopleCode API Reference, “XmlDoc Class”

Enterprise PeopleTools 8.45 PeopleBook: PeopleSoft Application Engine, “Tracing Application Engine Programs”

Enterprise PeopleTools 8.45 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 *xs/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 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 XmlDocument object and process the data using the XmlDocument 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 12, “Applying Filtering, Transformation and Translation,” Filtering Messages, page 255.
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.

Working with Message Samples

When developing a transform program for XML message data, you might find it useful to have sample copies of the initial and resulting XML message structures as a guide. You can generate the samples from the message definitions using the Create Test Message feature of Application Designer.

After you publish a test message containing sample data, you can view its XML structure and content in Integration Broker Monitor and copy that data to a text file for reference.

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]]>
</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.

See Also

[Chapter 8, "Sending and Receiving Messages," Generating Test Messages, page 192](#)

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">
  <!-- 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 XmlDocument object. However, you may want to transform the message using the PeopleCode Rowset class. Because XmlDocument object structure is compatible with Rowset object structure, you can copy the XML data to a rowset using the XmlDocument 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 XmlDocument 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 XmlDocument &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. . . */

/* Copy transformed rowset back to XmlDocument object */
&OK = &tempDoc.CopyRowset(&TempRS, "MY_MSG_NAME", "MY_MSG_VERSION");

```

See Also

Enterprise PeopleTools 8.45 PeopleBook: PeopleCode API Reference, “XmlDoc Class,” XmlDoc Methods

Filtering Messages

This section provides an overview of message filtering and discusses how to work with a PeopleCode filtering example.

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 XmlDocument 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">
    <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;
        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 5, “Understanding Supported Message Structures,” page 57](#)

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">
```

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/>

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 result message should contain. These name/value pairs are known as <i>match names</i> and <i>match values</i>. 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 <i>return value</i>. 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, Codeset Groups.

Code Set Groups

Code Set Group Name CS_SAP_01

Customize | Find | First 1-6 of 6 Last

	Match Name	Match Value	
1	business	US_mail	+ -
2	cardtype	visa	+ -
3	locale	en_us	+ -
4	type	ground	+ -
5	type	priority	+ -
6	uom	box	+ -

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.

- 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 9, “Configuring Nodes and Transactions,” Configuring Nodes, page 196

Defining Codesets

Use the Codesets page in the Codesets component (IB_CODESET). Select PeopleTools, Integration Broker, Codesets to access the Codeset page.

Code Sets

Code Set Group Name CS_SAP_01

Code Set Name PS_SAP_PO_01

	*Match Name	Match Value
1	locale	
2	locale	en_us
3	uom	
4	uom	box

Codeset page

To define a codeset:

- Add a new value and enter a codeset group name on which to base this codeset.
- 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.

- Add a new row.
- Select a match name from the set defined for the associated codeset group.
- 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.

- 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, Codeset Values to access the Codeset Values page.

Code Set Values

From Group CS_SAP_01

Code Set Name PS_SAP_PO_01

To Group CS_PSFT_01

Code Sets

Find | View All

First 1 of 1 Last

Description

SHIPMETHOD_SAP_PS_01

+ -

Select	Match Name	Match Value
☐	locale	
☒	locale	en_us
☐	uom	
☒	uom	box

Code Set Values

Customize | Find | View All

First 1-4 of 4 Last

*Return Name	Return Value
PS_LOCALE_01	en_uk + -
PS_UOM_01	carton + -

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.

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.

Deleting Codeset Values

To delete codeset values for a codeset:

1. Select PeopleTools, Integration Broker, 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, 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 <i>codeset</i> .
codesetname	Identifies the codeset whose name/value permutations you want to match in the input data. The transaction modifier that invokes this transform program identifies the initial and result nodes involved, and PeopleSoft Integration Broker examines their definitions to determine the <i>From</i> group and <i>To</i> 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 transaction modifier. PeopleSoft Integration Broker uses the specified node's codeset group as the <i>From</i> group key, thus invoking a different codeset values definition.
dest	(Optional) Overrides the name of the result node specified by the transaction modifier. PeopleSoft Integration Broker uses the specified node's codeset group as the <i>To</i> 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 transaction modifier; 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:

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 <i>yes</i> to ensure that the node specified by the <i>select</i> attribute is created if it does not exist yet. The return value is inserted as the value of that node.
createNodeFrom Value	(Optional) Set to <i>yes</i> to use the return value as the name of a new node, created where the <i>select</i> attribute specifies. The <i>value</i> tag can contain a valid XSLT value for that node, usually specified as an <i>xsl:value-of</i> 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 12, “Applying Filtering, Transformation and Translation,” Using XSLT for Transformation, page 258

<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:

```
<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">
  <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 emphasize 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>
        </carrier>
      </delivery>
    </dest>
  </xsl:template>

```

```

        </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 id="1">
        <name>pencil</name>
        <qty>15</qty>
        <uom>Carton</uom>
        <type>Bic</type>
    </li>
    <li id="2">
        <name>paper</name>
        <qty>10</qty>
        <uom>Box</uom>
        <type>Bic</type>
    </li>
</dest>
    <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*/
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 8.45 PeopleBook: PeopleCode Language Reference, “PeopleCode Built-in Functions,” FindCodeSetValues

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.

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 Integration Broker 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.

See Also

[Chapter 14, "Using Integration Broker Monitor," Monitoring Messaging System Information, page 304](#)

CHAPTER 13

Administering Relationships

This chapter discusses how to:

- Determine relationship parameters.
- Configure a relationship.
- Inactivate a relationship.
- Delete a relationship.
- Manage transaction modifiers.

Understanding Relationships

This section discusses when to use relationships and provides an overview of transaction modifiers.

Every integration requires at least one transaction at each PeopleSoft Integration Broker node. One node uses a transaction to send a message, and one or more nodes use transactions to receive the message. The sending node might apply a transaction with different parameters than those applied by the nodes that receive the message, with respect to routing, transmission type, message structure, or message content. A *relationship* reconciles incompatible parameters to successfully transmit data from the source to the target.

When to Use Relationships

You must define a relationship if any of the following is true:

- The source node sends a message asynchronously, but a target node expects to receive it synchronously.
- You're using a hub configuration to route messages.
- One or more messages in a transaction needs to be filtered, transformed, or translated upon sending or receiving.

Relationships *aren't* required for basic messaging if all of the following are true:

- Each node expects the message to have the same structure, version, and encoding as the other nodes.
- The transactions defined at each participating node all specify the same parameters (except transmission direction, which depends on a node's point of reference).
- You're not using a hub configuration (messages are sent directly from the source node to the target nodes).

Note. The relationship definitions you create, and the node definitions, transaction definitions, and transform programs they reference, are stored in the database you're signed on to, which is the default local node.

See Also

[Chapter 12, “Applying Filtering, Transformation and Translation,” page 247](#)

[Appendix A, “Integration Scenarios,” page 421](#)

Understanding Transaction Modifiers

A relationship applies a *transaction modifier* to an initial transaction, which produces a resulting transaction that’s appropriate for the target node. Only certain combinations of initial and resulting transaction types are supported, and both transactions must already be defined.

The following transaction type combinations are supported by PeopleSoft Integration Broker relationships.

Transaction Type Combination	Purpose
InAsync to InSync	Redirects an inbound asynchronous request message to a synchronous request handler at the target node.
OutAsync to OutSync	Routes an outbound asynchronous request message at a non-hub node to a target node that supports only synchronous requests. Note. This combination isn’t supported when both the initial and resulting transactions specify the default local node.
InAsync to OutSync	At a hub node, routes an inbound asynchronous request message to a target node that supports only synchronous requests.
InAsync to OutAsync	At a hub node, routes an inbound asynchronous request message to a target node.
InSync to OutSync	At a hub node, routes an inbound synchronous request message to a target node.
OutAsync to OutAsync	Applies a transform program to an outbound asynchronous request message at a non-hub node.
OutSync to OutSync	Applies a transform program to an outbound synchronous request message at a non-hub node.
InAsync to InAsync	Applies a transform program to an inbound asynchronous request message.
InSync to InSync	Applies a transform program to an inbound synchronous request message.

Note. An outbound to inbound modifier would essentially define a local integration, in which the default local node is both source and target. However, unlike cross-node messaging, you need to define only one transaction for a local integration, and PeopleSoft Integration Broker automatically routes the message appropriately, so no transaction modifier is necessary.

See Also

[Chapter 9, “Configuring Nodes and Transactions,” Configuring Transactions, page 204](#)

[Chapter 12, “Applying Filtering, Transformation and Translation,” page 247](#)

[Appendix A, “Integration Scenarios,” page 421](#)

Determining Relationship Parameters

This section discusses how to:

- Select a transaction type combination.
- Determine where to define the relationship.

Selecting a Transaction Type Combination

The transaction type combination that you apply with a transaction modifier depends on the difference between what the source node sends and what the target node expects. The following scenarios can be handled using relationships:

- Different transmission types.
- Hub routing.
- Filtering, transformation, or translation.

The source or target node for any of these scenarios can itself be a hub. Each transaction modifier applies to a single segment of the journey a message makes from the initial source node to the ultimate target node. Additionally, any transaction modifier can invoke a transform program.

Different Transmission Types

In this scenario the source node sends an asynchronous request message, but the target node expects a synchronous message. Choose from the following solutions:

- The target node converts the inbound asynchronous message to synchronous using an InAsync to InSync modifier.
- The source node converts the outbound asynchronous message to synchronous before sending it using an OutAsync to OutSync modifier.
- A hub node receives the inbound asynchronous message and converts it to an outbound synchronous message using an InAsync to OutSync modifier.

Hub Routing

In this scenario you’re using the default local node as a PeopleSoft Integration Broker hub, and it receives an inbound request message. Choose from the following solutions:

- For an asynchronous message, the hub routes it to the target node using an InAsync to OutAsync modifier.
- For an asynchronous message, the hub converts it to an outbound synchronous message and routes it to the target node using an InAsync to OutSync modifier.
- For a synchronous message, the hub routes it to the target node using an InSync to OutSync modifier.

Note. No special setup is required to use a hub configuration. Any node where you apply an inbound to outbound transaction modifier is by definition a hub node. You can treat any PeopleSoft Integration Broker node simultaneously as a hub node and as a non-hub node for the same transaction.

See [Chapter 13, “Administering Relationships,” Retaining Messages at Hub Nodes, page 288.](#)

Filtering, Transformation, or Translation

The target node expects a different request message or request message version than the source node sends, or the source node expects a different response message or response message version than the target node sends. Choose from the following solutions:

- The source node applies a transform program to the outbound asynchronous message using an OutAsync to OutAsync modifier.
- The source node applies a transform program to the outbound synchronous message using an OutSync to OutSync modifier.
- The target node applies a transform program to the inbound asynchronous message using an InAsync to InAsync modifier.
- The target node applies a transform program to the inbound synchronous message using an InSync to InSync modifier.

See Also

[Chapter 12, “Applying Filtering, Transformation and Translation,” page 247](#)

Determining Where to Define the Relationship

You define a relationship at the node that will change the transmission type, apply the routing or invoke the transform program specified by a transaction modifier. Your choice depends on which node is best suited for the solution you need to apply.

If you need more than one type of solution, you might be able to combine them in a single relationship, or you might need to define separate relationships at different nodes. Consider which nodes you have control over, or whether you’re using a hub configuration.

The following example implements all three solution types with three different relationships:

1. At the initial source node, change the transmission type using an OutAsync to OutSync modifier.
2. At the hub, route the message using an InSync to OutSync modifier.
3. At the ultimate target node, transform the message using an InSync to InSync modifier.

The following example implements all three solution types with a single relationship:

At the source node, use a single OutAsync to OutSync modifier to change the transmission type, transform the message, and route the message directly to a different target node.

Note. Only one transaction modifier is applied to a given message at any node. Multiple transaction modifiers can't be applied in sequence to a single message at a single node.

Configuring Relationships

A relationship applies a transaction modifier, which modifies the transmission type, destination, or message provided by the initial transaction, and uses the resulting transaction to pass the message on to its target. .

This section discusses how to:

- Define a relationship.
- Override node properties.
- Override contact information.

Understanding Configuring Relationships

To configure relationships, use the Relationships component (IB_RELATIONSHIP). To access the component, select PeopleTools, Integration Broker, Relationships

Defining Relationships

Select PeopleTools, Integration Broker, Relationships.

The screenshot shows the 'Relationships' page in PeopleTools. At the top, there are two tabs: 'Relationships' (selected) and 'Transaction Modifiers'. Below the tabs, the 'Relationship ID' is 'QE_SYNC_RELATIONSHIP'. A dark bar labeled 'Details' is followed by a text field for '*Description' containing 'Sync Transformation'. Below that is a dropdown for '*Relationship Status' set to 'Active'. Another dark bar labeled 'Node' is followed by a text field for '*Node Name' containing 'QE_IBTGT', with links for 'Properties' and 'Contact'. This structure is repeated for a second node. At the bottom is a large text area for 'Comment'.

Relationships page

Note that the Transactions Modifiers tab does not appear on until you create a relationship definition and save it.

Description	Enter a descriptive name for the relationship.
Relationship Status	Select from: • <i>Active</i>: All of this relationship's transaction modifiers are eligible to be applied by the integration engine. This is the default value. • <i>Inactive</i>: None of this relationship's transaction modifiers are eligible to be applied by the integration engine.
Node Name	Select the names of the two nodes to which this relationship applies. Only transactions included in the selected nodes' definitions can be used by this relationship. You can specify the same node name twice, which still constitutes a pair as far as a relationship is concerned. If you select the same node in both fields, you can use the relationship for both inbound-to-inbound and outbound-to-outbound transaction type combinations — the node you specify can be either the source or the target. If you select a different node in each field, you can use the relationship for inbound-to-outbound (hub) transaction type combinations going in either direction — either node can be the source of the inbound request message, with the other as the target of the outbound request message. Note. You can define only one relationship for each node pair. Never enter the name of the default local node.
Properties	Click Properties to access the Relationship Properties page and override the original properties of each node for this relationship.
Contact	Click Contact to access the Relationship Contact page and override the original contact information of each node for this relationship.

Activating Relationships

When a relationship is active, PeopleSoft Integration Broker can invoke it to apply its transaction modifiers to qualifying transactions. However, the relevant message and node definitions in the local database must both be active as well. These elements have the following dependencies:

- When you deactivate a message, all associated relationships are deactivated.
- When you deactivate a node, all associated relationships are deactivated.
- You can specify only active nodes when defining a new relationship.
- You can activate any existing relationship without restriction.

Note. If you deactivate a node or a message, all relationships associated with it are also deactivated. Activating the node and the message won't automatically reactivate the relationships; they must be individually reactivated.

Deactivating a relationship has no effect on the active status of other Integration Broker elements, but they do take it into account:

- If the integration engine invokes a transaction that's associated with an inactive relationship definition, the relationship is ignored and the transaction runs as defined for its associated node.
- When you change the status of a relationship definition — either from active to inactive or from inactive to active — all elements of Integration Broker receive notification of the change in

status, and the integration engine properly takes the relationship's new status into account when invoking transactions and transaction modifiers associated with it.

See Also

[Chapter 13, “Administering Relationships,” Editing Transaction Modifier Details, page 284](#)

Overriding Node Properties

Click the Properties link next to a node name on the Relationships page to access the Relationship Properties page.

Relationship Properties

Relationship ID QE_SYNC_RELATIONSHIP

Node Name QE_IBTGT

Properties				Customize	Find	First	1 of 1	Last
	*Name Type	*Property Name	Value	Comment				
1	Category	Greeting	Hello					

Relationship Properties page

Use this page to override the original properties of each node for this relationship. You can use these properties to update messages with additional information, such as adding a specific customer ID that's needed by a trading partner when you send orders. Properties created for all relationships are stored in a single table, PSRELATIONPROP.

This page works the same way as the Node Definitions — Properties page.

See [Chapter 9, “Configuring Nodes and Transactions,” Defining Node Properties, page 201](#).

Note. These properties are saved as part of the relationship. Be sure to save the relationship after you edit this page.

Overriding Contact Information

Click the Contact link on the Relationships page to access the Relationship Contact page.

Relationship Contact

Relationship ID QE_SYNC_RELATIONSHIP

Node Name QE_JBTGT

Contact

Company ID

Contact Manager

Contact Email

Contact Phone Number

Relationship Contact page

Use this page to override the original contact information of each node for this relationship. This page works the same way as the Node Definitions — Contact/Notes page.

See [Chapter 9, “Configuring Nodes and Transactions,” Specifying Contact Information, page 201](#).

Note. This contact information is saved as part of the relationship. Be sure to save the relationship after you edit this page.

Inactivating Relationships

To inactivate a relationship:

1. Select PeopleTools, Integration Broker, Relationships.
2. Select the relationship to inactivate. The Nodes page appears.
3. From the Status drop-down list, select Inactive.
4. Click the Save button.

Deleting Relationships

You cannot delete relationships. As an alternative, you can inactivate a relationship and define a new one.

See Also

[Chapter 13, “Administering Relationships,” Inactivating Relationships, page 280](#)

[Chapter 13, “Administering Relationships,” Defining Relationships, page 277](#)

Managing Transaction Modifiers

This section discusses how to:

- View transaction modifiers.
- Define transaction modifiers.
- Edit transaction modifier details.
- Apply asynchronous to synchronous transaction modifiers.
- Retain messages hub nodes.
- Inactivate transaction modifiers.
- Delete transaction modifiers.

Viewing Transaction Modifiers

Select PeopleTools, Integration Broker, Relationships, Transaction Modifiers to access the Relationships — Transaction Modifiers page. This page displays all transaction modifiers defined for the current relationship.

Relationships **Transaction Modifiers**

Relationship ID QE_ASYNC_RELATIONSHIP

Node Name QE_IBTGT

Node Name QE_IBTGT

Transaction Modifiers						Customize Find	First 1-3 of 3 Last
Initial	Result						
Initial Node	Transaction Type	Request Message Name	Effective Date	Status			
1 Edit QE_IBTGT	OA	QE_FLIGHTPLAN_TRANSFORM	09/08/2003	Active			
2 Edit QE_IBTGT	OA	QE_ASYNC_FTR_MSG	05/30/2003	Active			
3 Edit QE_IBTGT	OA	QE_ASYNC_MSG	05/30/2003	Active			

[Add Transaction Modifier](#)

Relationships — Transaction Modifiers page

Edit Click to access the Transaction Modifier Detail page and modify transaction modifier properties.

Add Transaction Modifier Click to access the Relationship Transactions page and define a new transaction modifier.

Defining Transaction Modifiers

Click Add Transaction Modifier on the Trans Modifiers page to access the Relationship Transactions page.

Relationship Transactions

Find an Existing Value
Add a New Value

Relationship ID:	QE_SYNC_RELATIONSHIP
Effective Date:	03/10/2004
Initial Node:	
Request Message Name:	
Source Request Message Version:	
Transaction Type:	
Result Node:	
Request Message Name:	
Target Request Message Version:	

Relationship Transactions page

Relationship ID

The ID for the current relationship is already entered, but you can select a different ID with which to associate this transaction modifier.

Note. If you enter a different relationship ID than the current one, the selected relationship's definition opens when you save the transaction modifier.

Initial Node and Result Node

Enter the name of the first or initial node that is sending or receiving the message.

Request Message Name

For both the initial and the resulting transaction, select the name of the request message each transaction should transmit.

Note. Both request messages you select must be in the same channel.

Source Request Message Version

Enter the message version for the request message that the initial node sends or receives.

Transaction Type

For the initial transaction, select from:

- *Outbound Asynchronous:* The default local node sends a request message to the selected node.
- *Outbound Asynchronous:* The default local node sends a request message to the selected node.
- *Outbound Synchronous:* The default local node sends a request message to the selected node, requiring a response.
- *Inbound Asynchronous:* The default local node receives a request message from the selected node.
- *Inbound Synchronous:* The default local node receives a request message from the selected node, requiring a response.

Result Node	Enter the name of the result or second node that receives or sends a message to the initial node.
Target Request Message Version	Enter the message version for the request message that the result node sends or receives.
Add	Click to confirm the parameters you entered for the new transaction modifier and access the Transaction Modifier Detail page.

Only transactions included in the selected node definitions can be used by this relationship. As you select each parameter for a transaction (node, transaction type, request message name, request message version), the available choices for the remaining parameters are reduced to those that combine with your current choices to define existing transactions. For example, if you select a node with only inbound asynchronous transactions defined, Inbound Asynchronous is the only transaction type available.

Important! The integration engine should not be presented with two transaction modifiers that have initial transactions from the same node definition, specifying the same message and the same transaction type. The integration engine will attempt to apply both modifiers in parallel, producing two resulting transactions that send different results to the same target node. The effects of this situation are unpredictable, and it should be avoided.

Make sure you don't select the same initial transaction for any two transaction modifiers in the same relationship. Also, if you select the same initial transaction for two transaction modifiers in different relationships, make sure the result transaction doesn't target the same node in both modifiers.

Connector Overrides in Transaction Modifiers

When you define a transaction modifier, be aware of the way it handles connector overrides specified by the transactions involved. In some cases, the modifier ignores all connector overrides and uses the connector and its properties specified by the node associated with the result transaction.

A connector override is applied in a transaction modifier only if both of the following are true:

- The connector override is specified by the result transaction.
- A connector override is *not* specified by the initial transaction.

Note. This behavior must be taken into account if you use a transaction both in basic integrations and in a transaction modifier, because the connector override is applied with the basic integration, but might not be applied in the transaction modifier.

For example:

1. Define node X containing transaction X. The node specifies connector A, so the transaction uses connector A for basic integrations.
2. Define node Y containing transaction Y. The node specifies connector B, so the transaction uses connector B for basic integrations.
3. Define a transaction modifier with initial transaction X and result transaction Y. The modifier will use connector B. However:
 - If you don't specify a connector override for initial transaction X, but specify connector O as an override for result transaction Y, the modifier will use connector O.
 - If you specify any connector override for initial transaction X, the modifier reverts to connector B, no matter what you've specified for result transaction Y.

Note. A connector override doesn't need to specify a different connector than the one specified by the node; it might just apply different properties for the same connector. The behavior described here applies to any type of connector override setting.

See Also

[Chapter 9, "Configuring Nodes and Transactions," Configuring Transactions, page 204](#)

Editing Transaction Modifier Details

Click Edit on the Transaction Modifiers page to access the Transaction Modifier Detail page. You can also click Add on the Relationship Transactions page.

Relationships

Transaction Modifiers

Relationship ID

QE_SYNC_RELATIONSHIP

Node Name

QE_IBTGT

Node Name

QE_IBTGT

Effective Date

05/30/2003

*Status

Active

Sequence number

Initial

Node

QE_IBTGT

Transaction Type

OS

Request Message Name

QE_SYNC_MSG

Version

VERSION_1

Result

Node

QE_IBTGT

*Transaction Type

OS

Request Message Name

QE_SYNC_MSG

Version

VERSION_1

Transformations

Request

QE_SYNC_TRN

Response

[Return to Transaction List](#)

Transaction Modifier Detail page

Note. The transaction modifier's basic parameters are read-only, and can be altered only by defining a new transaction modifier with different values. The *Async Reply Message* and *Async Reply Message Version* fields appear only if you selected an asynchronous initial transaction and a synchronous resulting transaction.

Status

Select from:

- *Active:* The current transaction modifier can be applied when the selected relationship is invoked. This is the default setting.
- *Inactive:* The current transaction modifier won't be applied when the selected relationship is invoked.

Sequence number	For future use.
Transaction Type	For the result transaction, select from: • <i>Outbound Asynchronous</i>: The default local node sends a request message to the selected node. • <i>Outbound Synchronous</i>: The default local node sends a request message to the selected node, requiring a response. • <i>Inbound Asynchronous</i>: The default local node receives a request message from the selected node. • <i>Inbound Synchronous</i>: The default local node receives a request message from the selected node, requiring a response.
Request	To apply a transform program to the request message, select the name of the transform program.
Response	Displays only when working with synchronous transaction types. To apply a transform program to a synchronous response message, select the name of the transform program. Note. For synchronous transactions, you may need to apply a transform program to both the request and response messages, although there's no requirement that they both be applied by the same node.
Async Reply Message	Displays only when working with asynchronous transaction types. Enter the name of the message to be asynchronously returned to the source node upon receipt of the synchronous response.
Async Reply Message Version	Displays only when working with asynchronous transaction types. Enter the version of the message to be asynchronously returned to the source node upon receipt of the synchronous response.
Return to Transaction List	Click to access the Trans Modifiers page for the selected relationship. Note. Be sure to save the transaction modifier before returning to the transaction modifier list. If you don't, your changes will be canceled.

When PeopleSoft Integration Broker invokes a transaction whose parameters match those of the initial transaction specified in this transaction modifier, the transaction modifier is applied to produce and invoke the specified resulting transaction. A given transaction modifier is applied only if both it and its parent relationship are active.

Activating Transaction Modifiers

Transaction modifiers are dependent on their parent relationships in the following ways:

- When you deactivate a relationship, all associated transaction modifiers are deactivated.
- You can create new transaction modifiers for inactive transactions, but they can be saved only with inactive status.

Note. Activating the relationship or both transactions won't automatically reactivate the associated transaction modifiers; they must be individually reactivated.

Applying Asynchronous-to-Synchronous Modifiers

When you apply an asynchronous to synchronous transaction modifier, the request message is received by the target node in synchronous mode, so that node sends a synchronous response. The only way the source node can accept a response message is in asynchronous mode, so the transaction modifier converts the synchronous response into an asynchronous request and sends it to the source node as an independent asynchronous transaction in reply to the original transaction. You specify this request with the Async Reply Message and Async Reply Message Version fields on the Transaction Modifier Detail page.

Note. The message you specify must have the same structure as the response message sent by the target node, and both must be in the same channel as the initial request message.

You can define and apply the transaction modifier at either the source node or the target node. Because the asynchronous response message is technically unrelated to the initial asynchronous request, you must define appropriate transactions at both the source and target nodes to accommodate it. The following examples describe the transactions required for an exchange in which node A sends message X to node B and receives message Y in response.

Applying Transaction Modifiers at Source Nodes

In the node A database, an OutAsync to OutSync modifier specifying asynchronous “response” message MSG_Y (really just another request) is defined as part of the NODE_B to NODE_B relationship. To apply the modifier, you must have defined the transactions described in the following table.

Database	Node Definition	Transaction Type	Message	Comment
A	NODE_B	OutAsync	MSG_X	Source sends asynchronous request.
A	NODE_B	OutSync ¹	MSG_X, MSG_Y	Modifier at source produces and sends synchronous request. ¹ This is a single transaction, listed twice.
B	NODE_A	InSync	MSG_X, MSG_Y	Target receives synchronous request and sends synchronous response.
A	NODE_B	OutSync ¹	MSG_X, MSG_Y	Modifier at source receives synchronous response and produces asynchronous “response.” ¹ This is a single transaction, listed twice.
A	NODE_B	InAsync	MSG_Y	Source receives asynchronous “response.”

Applying Transaction Modifier at Target Nodes

In the node B database, an InAsync to InSync modifier specifying asynchronous “response” message MSG_Y (really just another request) is defined as part of the NODE_A to NODE_A relationship. To apply the modifier, you must have defined the transactions described in the following table.

Database	Node Definition	Transaction Type	Messages	Comment
A	NODE_B	OutAsync	MSG_X	Source sends asynchronous request.
B	NODE_A	InAsync	MSG_X	Modifier at target receives asynchronous request.
B	NODE_A	InSync	MSG_X, MSG_Y	Modifier at target produces synchronous request. Target receives synchronous request and sends synchronous response. Modifier produces asynchronous “response.”
A	NODE_B	InAsync	MSG_Y	Source receives asynchronous “response.”

Retaining Messages at Hub Nodes

If you’re using an inbound to outbound transaction modifier for message routing at a hub node, you may want the hub node to retain (receive and process) the message locally as well. You define an additional transaction modifier to accomplish this. The following example specifies the transactions required for an exchange in which source node A sends message X to hub node B, which routes the message to target node C and retains the message itself as well.

Applying Basic Routing

In the node B database, an InAsync to OutAsync modifier is defined as part of the NODE_A to NODE_C relationship. To apply the modifier, you must have defined the transactions described in the following table.

Database	Node Definition	Transaction Type	Messages	Comment
A	NODE_B	OutAsync	MSG_X	Source sends asynchronous request.
B	NODE_A	InAsync	MSG_X	Modifier at hub receives asynchronous request.

Database	Node Definition	Transaction Type	Messages	Comment
B	NODE_C	OutAsync	MSG_X	Modifier at hub routes asynchronous request to target.
C	NODE_B	InAsync	MSG_X	Target receives asynchronous request.

Retaining Messages

For node B to receive and process the same message, the appropriate InAsync transaction is already defined in the node B database, as shown in the table. Additionally, the node B database must contain a NODE_A to NODE_A relationship which includes an InAsync to InAsync modifier. The modifier essentially routes the message back through the original transaction, at which point node B can receive and process it normally.

Note. PeopleSoft Integration Broker doesn't support applying multiple transaction modifiers in sequence to a single message. However, you can apply them in parallel, as in this example.

Inactivating Transaction Modifiers

To inactivate a transaction modifier:

1. Select PeopleTools, Integration Broker, Relationships.
2. Select the relationship that contains the transaction modifier to inactivate. The Nodes page appears.
3. Click the Trans Modifier tab.
4. In the Transaction Modifiers section, click the Edit link of the transaction with which to work.
5. From the Status drop-down list, select Inactive.
6. Click the Save button.

Deleting Transaction Modifiers

You cannot delete transaction modifiers. As an alternative, you can inactivate a transaction modifier and then define a new one.

See Also

[Chapter 13, “Administering Relationships,” Defining Transaction Modifiers, page 281](#)

[Chapter 13, “Administering Relationships,” Inactivating Transaction Modifiers, page 289](#)

CHAPTER 14

Using Integration Broker Monitor

This chapter discusses how to:

- Monitor message system information.
- Monitor asynchronous message details.
- View synchronous message details.
- View messaging system performance statistics.
- Run batch error notification processes.
- Run batch message archiving processes.
- Purge messaging tables.
- Use the Message Monitor component interface.

Understanding Integration Broker Monitor

This section lists prerequisites and common elements and discusses:

- Integration Broker Monitor features and components.
- Brokers, contractors, and queues.
- Message server processes.
- Dispatchers and handlers.
- Asynchronous message publication.
- Asynchronous message subscription.
- Synchronous message publication.
- Synchronous message subscription.
- Message status.
- Blocked and stalled channels

Prerequisites

Before using Integration Broker Monitor, set up security for it using permission lists. To set up permission lists, select PeopleTools, Security, Permissions & Roles, Permission Lists.

You use permission lists to grant access to the Integration Broker Monitor pages and to grant read and write access to message channels. Users can view messages only in channels to which they have read access. All edit actions require write access.

Common Elements Used in This Chapter

Publisher

Usually the user ID of the person who triggered the message publication.



Click the Download button to download output to a spreadsheet.

Integration Broker Monitor Features and Components

System administrators use Integration Broker Monitor to monitor asynchronous and synchronous message transaction information, status information, and other integration engine data.

Features

Integration Broker Monitor provides:

- Status on channels, nodes, and individual messages.
You can also view and edit message XML.
- Control and administration of domains that have publication and subscribe (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 messages.
- Batch processes for error notification and message archival.
- Queries to gather information about messaging system metadata without your having to access PeopleSoft Application Designer.

For example, you can perform queries to retrieve information about:

- Transactions for a request or response message.
- Messages that belong to each channel.
- Channels and messages in the local node.

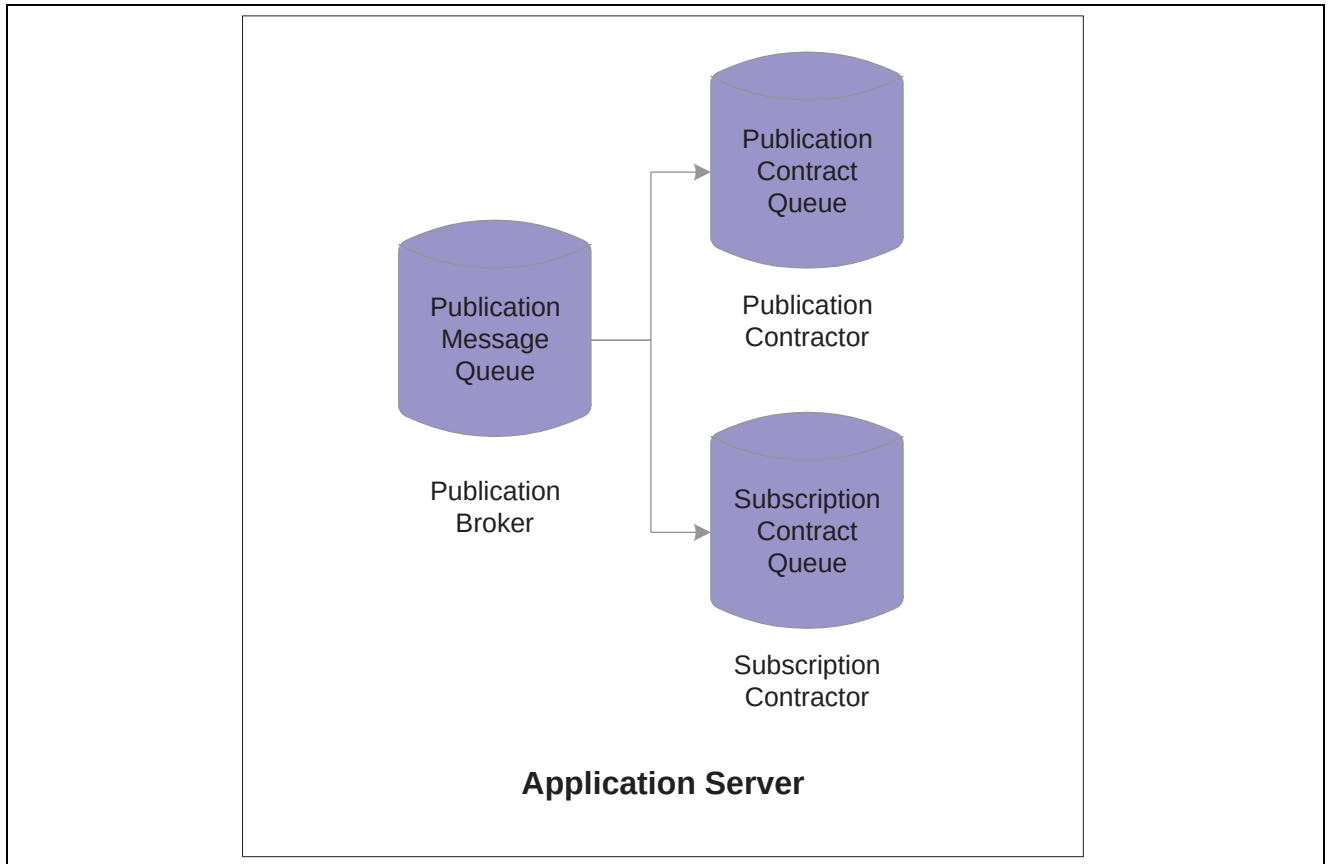
Components

There are five components associated with Integration Broker Monitor.

Monitor Message	View information about message instances, publication contracts, subscription contracts, synchronous messages, channel status, node status, and domain status. Generate a number of related predefined queries.
Message Details	View asynchronous message details, including information about the message instance, its publication or subscription contracts, error messages, and message instance XML. If transformations have been applied to the message, you can view the transformed XML for the publication and subscription contracts.
Synchronous Details	View information about synchronous message details and message 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 Messages	Run the batch process to archive messages.

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:



Brokers and contractors

Each broker or contractor has a queue to hold messages to process. The services perform the following functions:

Publication Broker

Acts as the routing mechanism. When an asynchronous message arrives in its queue, the Publication Broker service runs the defined routing rules. If the message needs to be published to a remote node, it routes the message to the Publication Contractor service. If the message is subscribed to on the local node, then the Publication Broker routes the message 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 message to the integration gateway. When the integration gateway sends a reply indicating that it received the publication message, 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 subscription PeopleCode. Then it updates the subscription contract concerning the status of the subscription processing.

Message Server Processes

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

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.

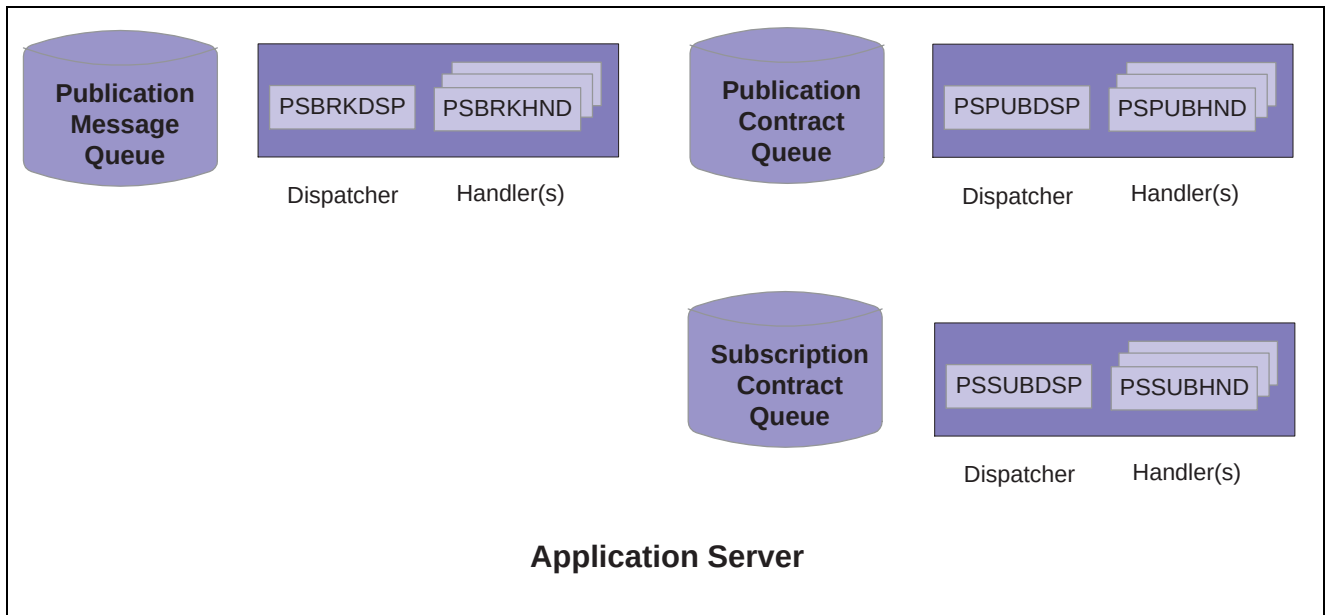
This relationship is analogous to the way that the application server handles workstation connections and requests. To handle the incoming client requests, the application server has a listener and a handler (or a pool of handlers). The listener receives the incoming requests and then routes them to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



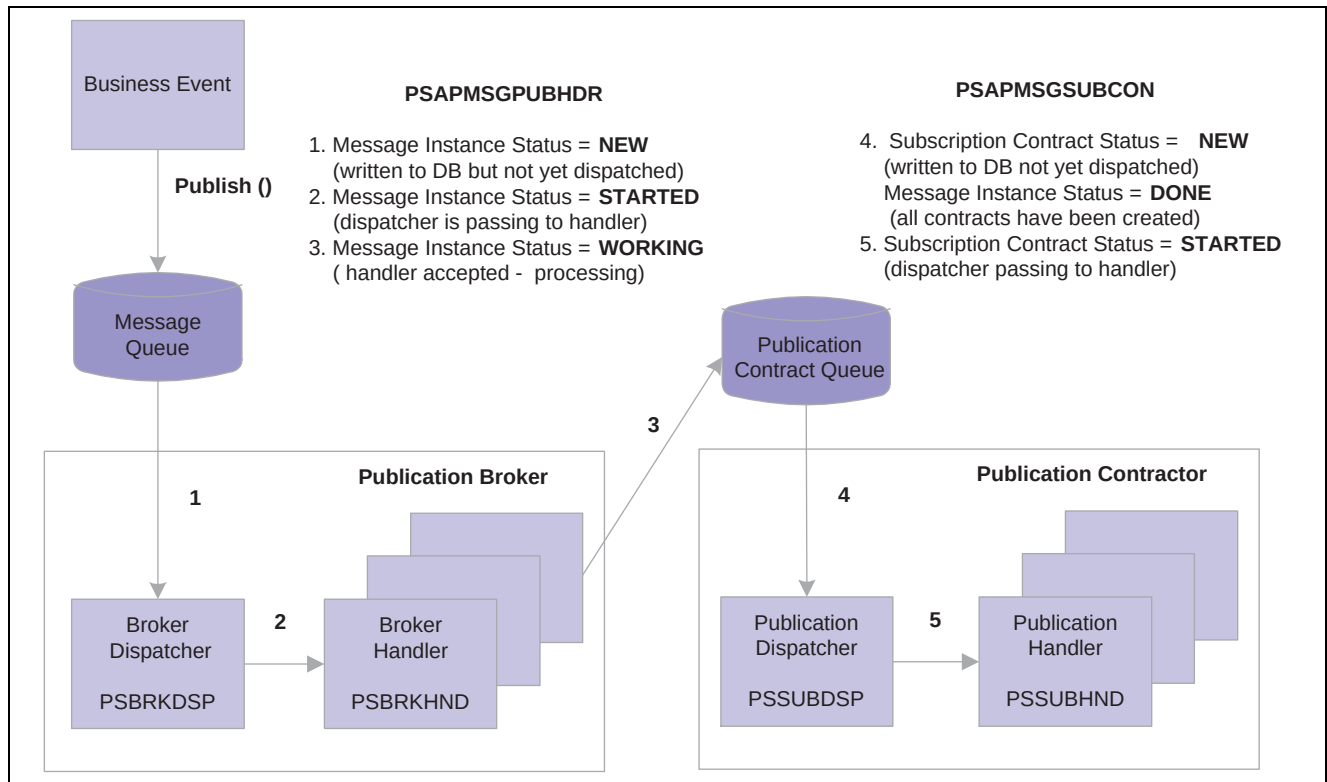
Dispatchers and handlers

Asynchronous Message Publication

This section describes the flow of an asynchronous message publication through PeopleSoft Integration Broker, as well as the status of the messages as they appear in Integration Broker Monitor.

Asynchronous Publish of a Message Instance

This diagram shows an asynchronous publish of a message instance:



Asynchronous publication of a message instance

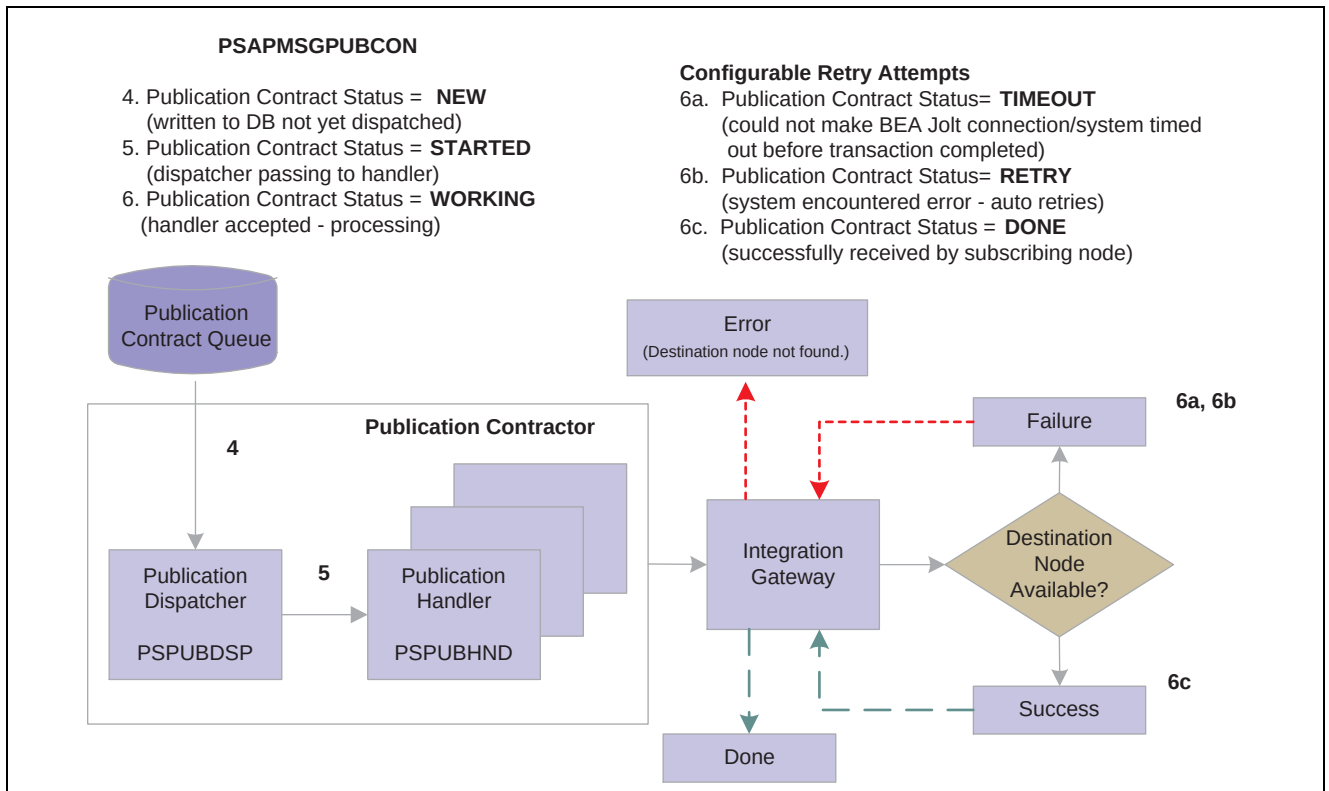
The processing steps of an asynchronous publication of a message instance are:

1. The message is published and enters the message queue.
The Broker Dispatcher process picks up the message instance from its queue. During this stage, the status of the message instance is *New*.
2. The Broker Dispatcher process passes the message instance to the Broker Handler process.
During this stage, the status of the message instance is *Started*.
3. The Broker Handler process accepts the message instance, reads the data, and runs the routing rules to determine where the publication needs to be delivered.
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 message instance is *Working*.
4. After the message is stored in the publication contract queue, the status of the publication contract is *New*, the message instance status is *Done*, and the Publication Dispatcher process picks up the publication contract from its queue.

You view message instance status on the Message Instance page of Integration Broker Monitor. To access this page, select PeopleTools, Integration Broker, Monitor, Monitor Message and select the Message Instance page.

Asynchronous Publish of a Publication Contract

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 message to the integration gateway.

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 messages have *Retry* status, the messages 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 message. The message goes back to the Publication Handler process and returns to *Working* status.

You can view the status information for the publication contract in Integration Broker Monitor by selecting PeopleTools, Integration Broker, Monitor, Monitor Message, Pub Contracts.

See Also

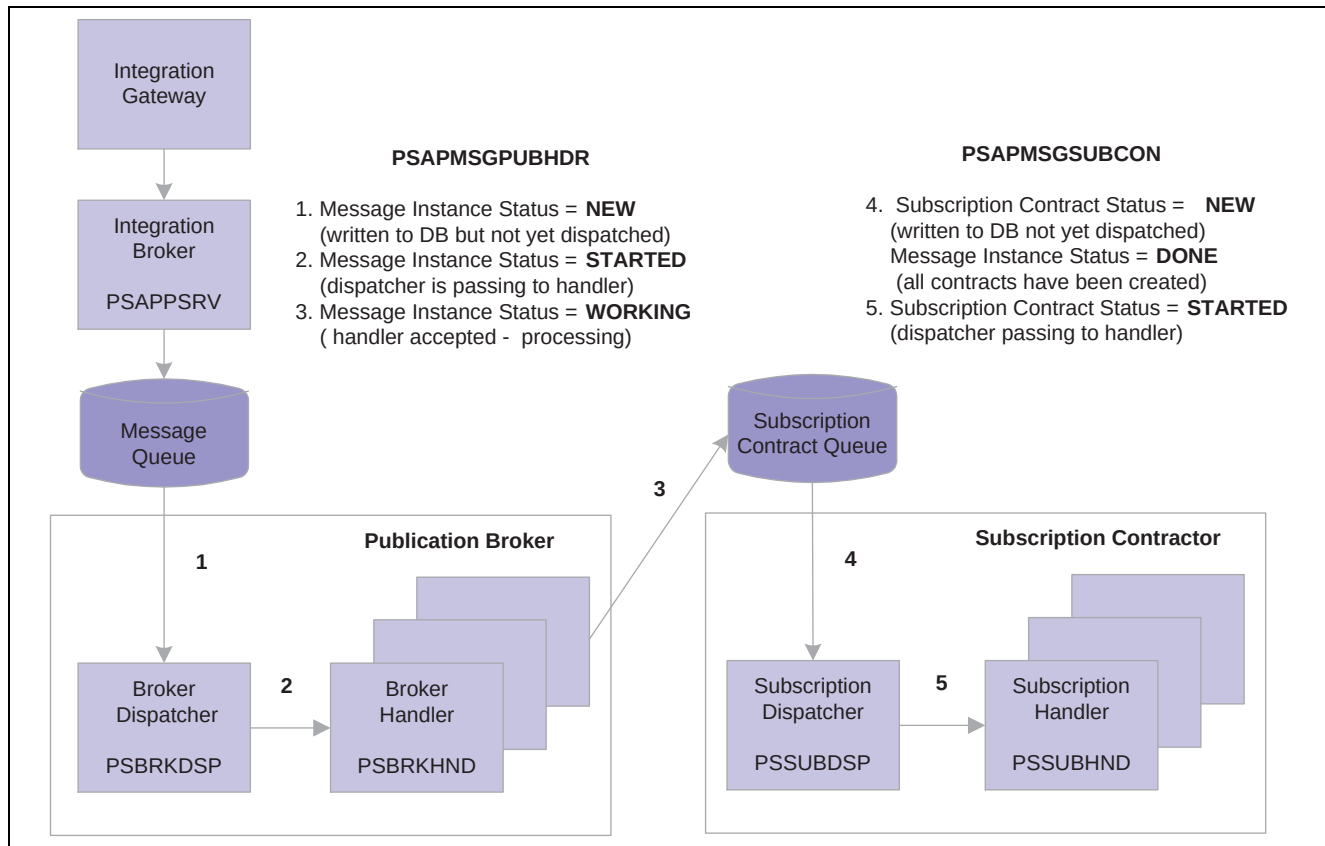
Chapter 14, "Using Integration Broker Monitor," Message Statuses, page 301

Asynchronous Message Subscription

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

Asynchronous Subscription of a Message Instance

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



Asynchronous subscription of a message instance

The processing steps are:

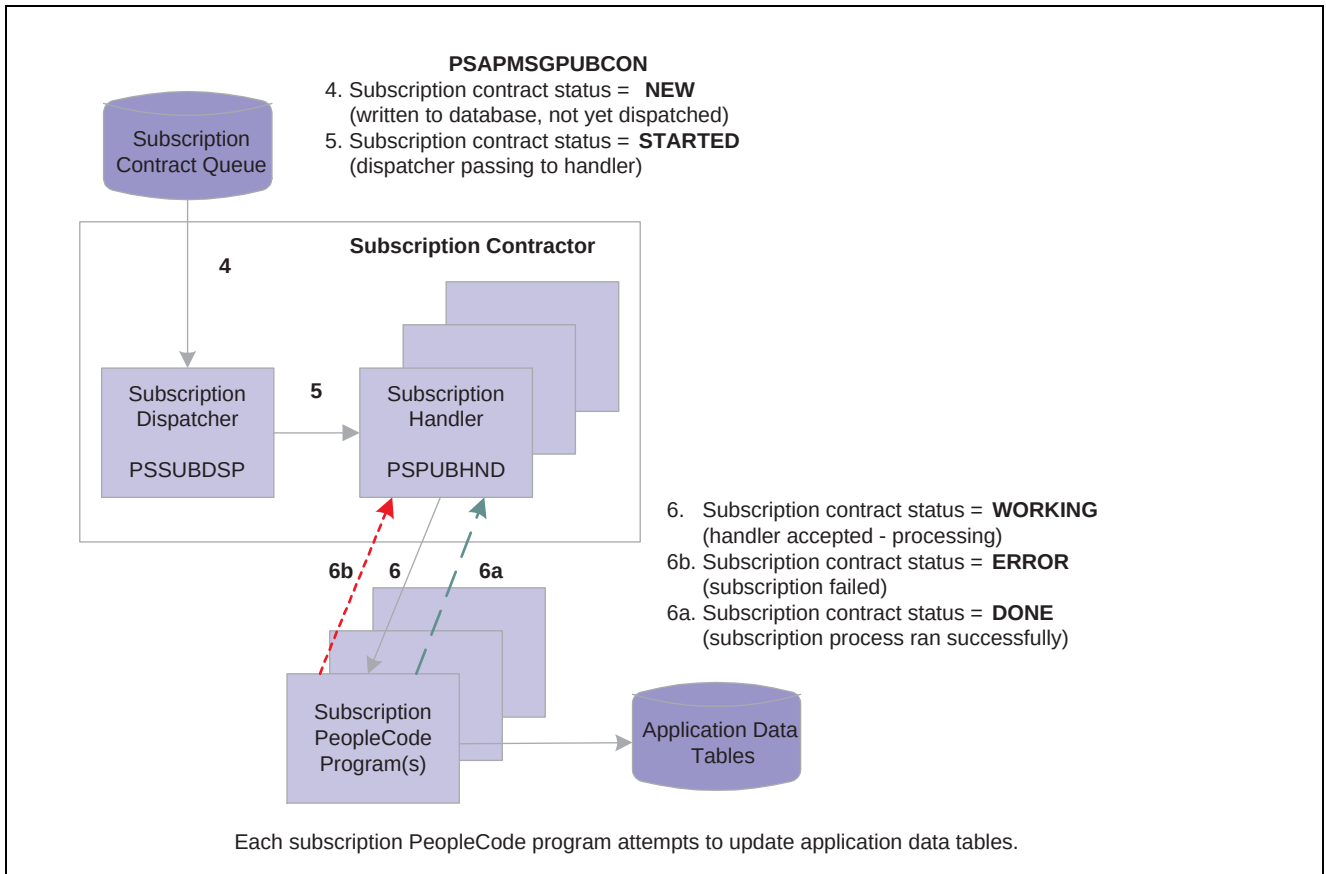
1. The message enters the message queue.
The Broker Dispatcher process picks up the message instance from its queue. During this stage, the status of the message instance is *New*.
2. The Broker Dispatcher process passes the message instance to the Broker Handler process.
During this stage, the status of the message instance is *Started*.
3. The Broker Handler process accepts the message instance, reads the data, and runs the subscription routing rules to determine if the message 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 message instance is *Working*.

Once the message is stored in the subscription contract queue, the status of the subscription contract is *New*, the message 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 message instance is *Done*, the subscription contract status is *New*.

You view message instance status on the Message Instance page of Integration Broker Monitor. To access this page, select PeopleTools, Integration Broker, Monitor, Monitor Message and select the Message Instance page.

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 subscription PeopleCode.

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

If the subscription 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).

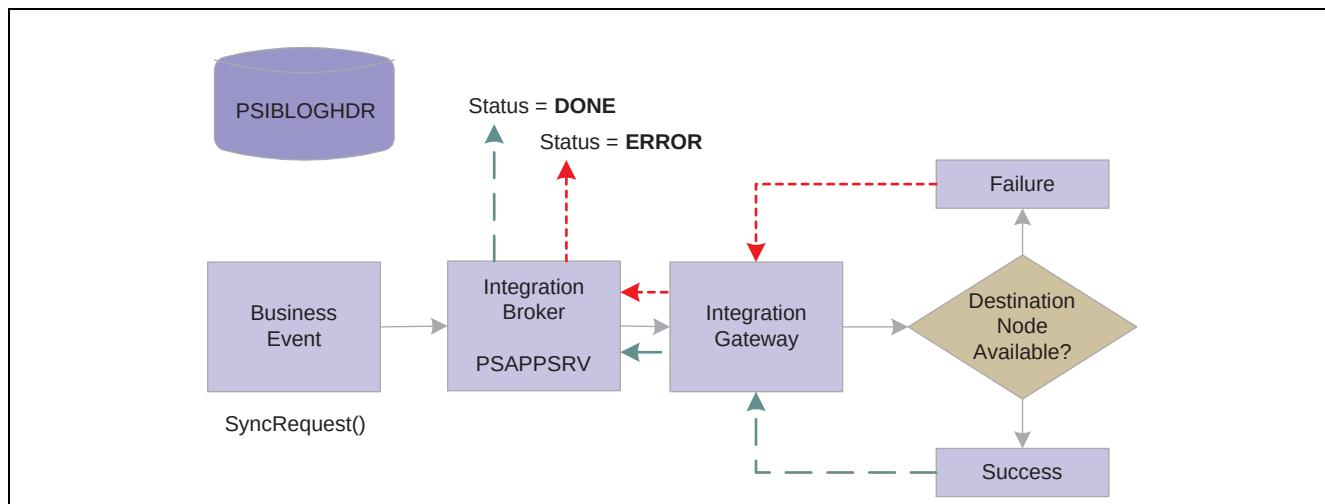
To view status information for subscription contracts, use the Sub Contracts page in Integration Broker Monitor. To access this page, select PeopleTools, Integration Broker, Monitor, Monitor Message, Sub Contracts.

See Also

[Chapter 14, “Using Integration Broker Monitor,” Message Statuses, page 301](#)

Synchronous Message Publication

This diagram illustrates synchronous message publishing through PeopleSoft Integration Broker:



Synchronous publication of a message instance

The processing steps are:

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

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

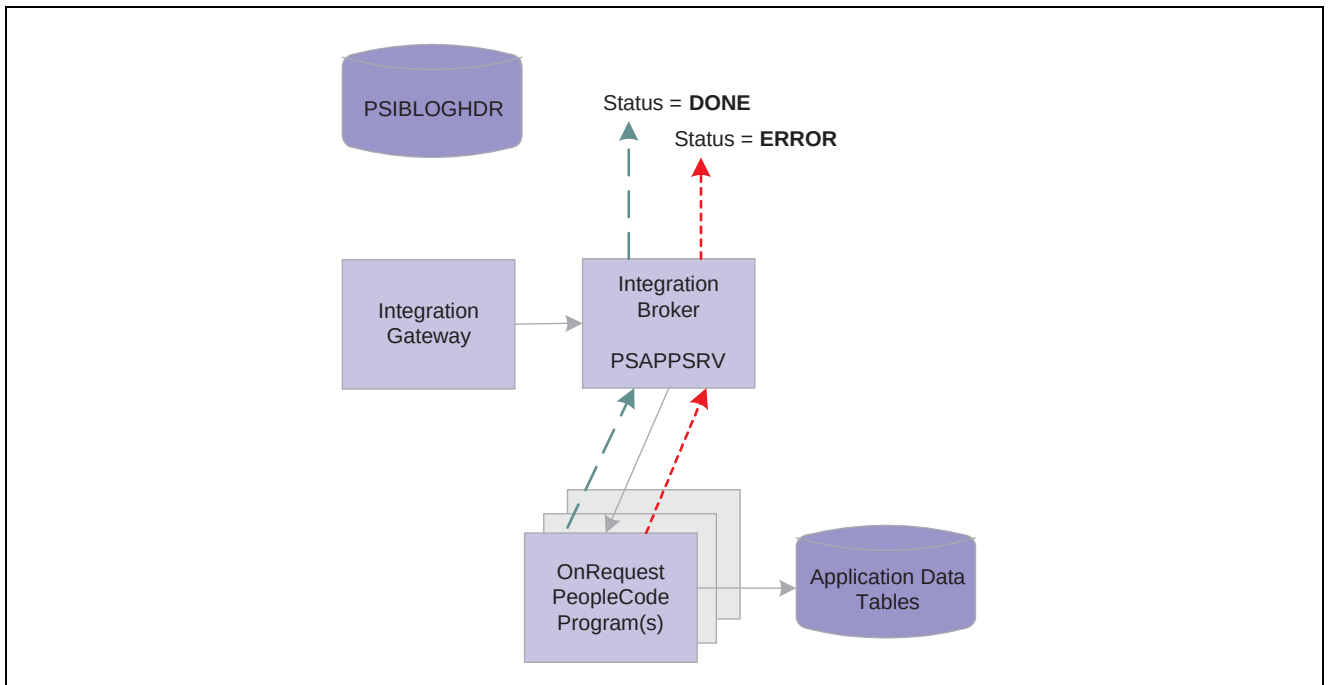
You can view the status information for the publication in Integration Broker Monitor using the Synchronous Messages page. Access this page by selecting PeopleTools, Integration Broker, Monitor, Monitor Message and selecting the Synchronous Messages page.

See Also

[Chapter 14, “Using Integration Broker Monitor,” Message Statuses, page 301](#)

Synchronous Message Subscription

This diagram illustrates synchronous message subscription through PeopleSoft Integration Broker:



Synchronous subscription of a message instance

The processing steps are:

1. The integration gateway passes an inbound synchronous message to the integration engine.
2. The integration engine runs a subscription PeopleCode program.
3. The subscription PeopleCode program attempts to update the application data tables.

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

You can view the status information for the publication in Integration Broker Monitor by using the Synchronous Messages page. Access this page by selecting PeopleTools, Integration Broker, Monitor, Monitor Message, and selecting the Synchronous Messages page.

See Also

[Chapter 14, “Using Integration Broker Monitor,” Message Statuses, page 301](#)

Message Statuses

This section discusses message status information that Integration Broker Monitor displays for synchronous and asynchronous messages.

Synchronous Message Statuses

For synchronous messages, Integration Broker Monitor displays one of two statuses.

Message Status	Description
<i>Done</i>	Indicates the synchronous request was successful.
<i>Error</i>	indicates that an error occurred during processing. Manual intervention is required.

Asynchronous Message Statuses

For asynchronous messages, Integration Broker Monitor displays different statuses as messages progress through the system.

The typical message status progression for asynchronous messages is:

1. New.
2. Started.
3. Working.
4. Contracts created.

However, Integration Broker Monitor can display any of the statuses listed in the following table.

Message Status	Description
<i>New</i>	Either the item has been written to the database but has not been dispatched yet, or the item has just been resubmitted.
<i>Started</i>	The dispatcher is in the process of passing the item to a handler, but the handler has not received it yet.
<i>Working</i>	The handler has accepted the item and is currently processing it.
<i>Contracts Created</i>	This status appears only for asynchronous messages. It indicates that the message instance has completed processing.
<i>Done</i>	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.
<i>Error</i>	An error occurred during processing. Manual intervention is required.
<i>Retry</i>	The system encountered an intermittent error during processing. The system retries messages with this status automatically.

Message Status	Description
<i>Timeout</i>	The system has reached the maximum retry count to send a message.
<i>Edited</i>	The publication data for the item has been edited. Processing does not resume until you resubmit the item.
<i>Cancelled</i>	The item has been canceled. The system cannot process the item until you resubmit it.

Blocked and Stalled Channels

This section discusses:

- Blocked channels.
- Stalled channels.

Blocked Channels

Blocked channels preserve the order in which messages are processed.

The pub/sub system guarantees that items are processed in the order they are sent. If a message has a status of *Error*, *Timeout*, or *Edited*, the message queue becomes blocked and no processing occurs until you resolve the problem with the message.

For publications, channels are partitioned in queues by subchannels.

For publication contracts, the channel is further partitioned into queues by subchannel and target node. If a channel is ordered, items in that channel 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 channel is unordered, an item (such as the publication, publication contract, or subscription contract) never blocks another item. All items are processed in parallel.

Stalled Channels

Stalls do not occur by design. They are caused by gaps in functionality, user errors, defects, and so forth.

For example, a channel 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.

Monitoring Messaging System Information

You can monitor message system queue information by using the Monitor Message component (APPMMSGMONITOR). The Monitor Message component enables you to view information about message instances, publication contracts, subscription contracts, synchronous messages, channel status, node status, and domain status. You can also use this component to generate related predefined queries.

This section discusses how to:

- Work with the Monitor Message component.
- Filter messaging information.
- View outputs.
- Monitor system queue information.
- Monitor asynchronous message instances.
- Monitor publication contracts.
- Monitor subscription contracts.
- Monitor synchronous message instances.
- Resubmit and cancel messages.
- View undelivered node transactions.
- Pause, test, and ping nodes.
- Monitor channel status.
- View channel partition information.
- Work with pub/sub server domains.
- Set up domain failover.
- View messaging system performance statistics.
- Run message queries.
- Archive message instances.

Working with the Monitor Message Component

This section discusses how to:

- Access the Monitor Message component.
- Use the Monitor Message component pages.

Accessing the Monitor Message Component

To access the Monitor Message component, select PeopleTools, Integration Broker, Monitor, Monitor Message. By default the Overview page is active.

The screenshot shows the 'Monitor Message' component interface. At the top, there are tabs: Overview, Message Instances, Pub Contracts, Sub Contracts, Synchronous Messages, Channel Status, and Node Status. The 'Overview' tab is selected. Below the tabs is a 'Message Criteria' section with filters: 'Publish Node' (QE_LOCAL), 'Last' (None), 'Archived' (checkbox), '*Queue Type' (Message Instance), and '*Group By' (Message). A 'Refresh' button is on the right. Below this is a 'Queues' section with a table. The table has columns: Message Name, Error, New, Started, Working, Done, Retry, Timeout, Edited, and Canceled. There are two rows of data: QE_F18_ASYNC and QE_F18_ASYNC_XMLDOC.

Message Name	Error	New	Started	Working	Done	Retry	Timeout	Edited	Canceled
QE_F18_ASYNC	0	0	0	0	9	0	0	0	0
QE_F18_ASYNC_XMLDOC	0	0	0	0	5	0	0	0	0

Monitor Message component

Using the Monitor Message Component Pages

The Monitor Message component consists of the following pages:

Overview	View the system queues at a high level so that you can isolate particular areas and then drill down for further information.
Message Instance	View all asynchronous messages from remote nodes or applications that publish information.
Pub Contracts	View outbound publication contracts that are sent to remote message nodes with which the system is interacting.
Sub Contracts	View work orders to run PeopleCode programs to which the local node subscribes.
Synchronous Messages	View inbound synchronous messages from remote nodes or applications that publish information.
Channel Status	View the status of the message channels defined in the system.
Node Status	View the status of the local message node. Ping remote nodes to determine their availability.
Domain Status	View the domains with pub/sub servers that are running against the application database of the local node. Pause the processing or take a domain offline. Force a reset, which places the dispatcher status for all processes on all machines in cleanup mode. Set domain failover for individual domains or for domain failover groups.
Statistics	View runtime performance statistics for asynchronous and synchronous messages that flow through the messaging system. View statistics in numeric or graphical format.
Queries	Access a predefined set of queries (built with PeopleSoft Query) that enable you to view details and generate reports about the message setup, message channels, and message nodes in your system.

Filtering Messaging Information

Before you begin monitoring the messaging system, there are a few general guidelines that enable you to quickly drill down to the information you need.

Because the Integration Broker Monitor provides information about the entire messaging system, you need to understand how to filter the information to reduce the number of items. For instance, rather than sifting through every message in the entire system, the Integration Broker Monitor enables you to sort by publish node, publish date and time, live and archived messages, and so on.

You can filter the result set on the following pages of the Monitor Message component:

- Overview page
- Message Instances page
- Pub Contracts page
- Sub Contracts page
- Synchronous Messages page

You can use the following filter criteria on each page where filtering applies:

Publish Node	Every message is stamped with the node that publishes it. The Publish Node drop-down list box contains all the nodes defined in the system.
	Note. Integration Broker Monitor only allows you to view information for the local system (database). However, the queues for the local database can contain messages published by remote nodes, as well the local node. There is only one local node for a database.
Last	Use this field to narrow or broaden the messages to display based on intervals of time. Enter a numerical value, and select Days, Hours, or Minutes from the associated drop-down list. Select None to view all messages currently in the system.
Archived	The Archived check box enables you to search for either archived or live message data. To search archived message data, select the check box. To search live message data, clear the check box.
Channel Name	To view message instances within a specific channel, select the appropriate channel value in the Channel Name drop-down list box.
Message Name	To view the instances of a particular message definition, select the appropriate message value from the Message Name drop-down list box.
Status	To view message instances by status, select the status criteria from the Status drop-down list box. The status options reflect the status columns that appear on the Overview page.

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.

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 that you want on the pages where filtering applies (namely, the Overview, Message Instances, Pub Contracts, and Sub Contracts pages).

2. Click Refresh on each page.

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 Integration Broker Monitor, the system displays the message data according to your most recent filtering selections.

Viewing Output

The output in the results grids of the Monitor Message component enables you to view information by channel, date, status, and so on.

To sort the information that appears in the results grids, click a column heading.

After you locate a particular message status, publication contract, subscription contract or other item, click the Details link to view detailed data associated with the item. The system launches the Message Details component (AMM_DETAILS), the Synchronous Details component (AMM_SYNCDETAIL), or an application-specific page where you can view additional data and correct errors.

Whether you use the Message Details component, Synchronous Details component, or an application-specific page to view and edit a message depends on the properties of the message definition in PeopleSoft Application Designer. The Details link invokes a transfer page to the component that you can specify in Application Designer by selecting Message Definition, Message Properties, Error Viewing/Correction.

Monitoring System Queue Information

To monitor system queue information, select PeopleTools, Integration Broker, Monitor, Monitor Message.

The Overview page is the first page that appears. Use this page for a high-level overview of the status of asynchronous message instances, publication contracts, and subscription contracts.

Depending on the information that you want, you then drill down further into the Message Instances page, the Pub Contracts page, and the Sub Contracts page.

Most of the time, the status for a message or channel that appears in the Queues scroll area is *Done*. This means that the message instance arrived in the publication queue (creating the message headers only). However, other statuses for a message or channel can appear. For instance, if the pub/sub system is down, the status is *New*. If there are transformation or OnRoute PeopleCode errors, the message status is *Error*. In addition, if you access Integration Broker Monitor at certain times, you might see a message status of *Started* or *Working*. Use the other pages in this component to view more comprehensive message status information.

In addition to using the filtering options discussed earlier in this chapter, you can filter the information on this page by using the following criteria:

Queue Type Specify a queue. Valid options are *Message Instance*, *Publication Contract*, and *Subscription Contract*.

Group By Select *Channel* or *Message*.

After you select your filtering options, click Refresh.

See Also

Chapter 14, “Using Integration Broker Monitor,” Working with the Monitor Message Component, page 304

Monitoring Asynchronous Message Instances

Select PeopleTools, Integration Broker, Monitor, Monitor Message, Message Instances to access the Message Instances page.

The Message Instances page enables you to monitor the status and details related to individual asynchronous message instances. Every publication and subscription contract is associated with a message instance.

After you select your filtering options, click Refresh.

Note. All asynchronous transactions are sent to the publication queue regardless of whether the transactions are active or inactive. As a result inactive transactions that appear on the Message Instances page do not contain any publication contracts.

See Also

[Chapter 14, “Using Integration Broker Monitor,” Working with the Monitor Message Component, page 304](#)

Monitoring Publication Contracts

Select PeopleTools, Integration Broker, Monitor, Monitor Message, Pub Contracts to access the Pub Contracts page.

This page shows outbound publication work orders to send to remote nodes. The system does not create publication contracts for routing to the local node.

After you select the filtering options, click Refresh to update the message criteria.

See Also

[Chapter 14, “Using Integration Broker Monitor,” Working with the Monitor Message Component, page 304](#)

Monitoring Subscription Contracts

Select PeopleTools, Integration Broker, Monitor, Monitor Message, Sub Contracts to access the Sub Contracts page.

The Sub Contracts page enables you to view work orders and run the PeopleCode programs to which the local node subscribes. Subscription contracts for remote nodes do not appear.

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 Last drop-down list box to select whether to filter information by the day, hour, or minute. In the box to the left of the drop-down list box, enter the number of days, hours, or minutes. The volume of messages in the system determines the best values to enter.

See Also

[Chapter 14, “Using Integration Broker Monitor,” Working with the Monitor Message Component, page 304](#)

Monitoring Synchronous Message Instances

Select PeopleTools, Integration Broker, Monitor, Monitor Message, Synchronous Messages to access the Synchronous Messages page.

The Synchronous Messages page displays inbound synchronous messages from remote nodes or applications that publish information.

See Also

[Chapter 14, “Using Integration Broker Monitor,” Working with the Monitor Message Component, page 304](#)

Resubmitting and Canceling Messages for Processing

The Message Instances page, Pub Contracts page, Sub Contracts page, and Synchronous Messages page of the Monitor Message component enable you to resubmit and cancel individual and bulk messages for processing.

Working with Individual Messages

To resubmit or cancel individual messages, select the check box next to the appropriate messages and click the Resubmit Message or Cancel Message button.

To deselect a message, clear the check box next to the message.

Working with Bulk Messages

In addition to the Clear All, Resubmit Message, and Cancel Message buttons, you also use the following link when working with bulk message.

Select All	Click to select all messages for resubmittal or cancellation. After you click this link, click the Resubmit Message or Cancel Message button as appropriate.

To deselect messages, click the Clear All button.

Viewing Undelivered Node Transactions

Integration Broker Monitor provides a Transaction Retry Queue link for read-only access to information about undelivered node transactions, such as the node name, transaction type, request message, and request message version.

You can access this information from the following pages in the Message Monitor component:

- Pub Contracts page.
- Node Status page.

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.

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.

The Transaction Retry Queue link appears on the Pub Contracts page and provides access to the Undelivered Node Transaction page. Use the Undelivered Node Transaction page for read-only access to information about undelivered node transactions.

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.

See Also

[Chapter 14, “Using Integration Broker Monitor,” Working with the Monitor Message Component, page 304](#)

Pausing, Testing, and Pinging Nodes

Select PeopleTools, Integration Broker, Monitor, Monitor Message, Node Status to access the Node Status page.

This section discusses how to:

- Add a pause time to a local node.
- Test local nodes.
- Ping remote nodes.

Adding Pause Times to Local Nodes

A pause time is an interval of time during which the message node becomes inactive. When the pause time begins, the message 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 message 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 message node that runs from 12 a.m. to 4 a.m.

During a pause time, messages are not published or received by the local system. When the system is paused, the node cannot accept messages sent to it. Consequently, the publishing node must attempt to send messages again later. The publishing node continues to send messages until it exceeds the local timeout period. When this happens, the message assumes a *Timeout* status in the publisher's message queue. The timeout period is an attribute of the publication channel, not the subscription channel.

If the system attempts to send a message while the node is paused, the system writes the message to the publication and subscription queues, but it cannot publish the message 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.

To add a node pause time:

1. Click Add Pause.
2. Select a day of the week in the Start Day drop-down list box.
3. Enter a value in the Start Time edit box.
4. Select a day of the week in the End Day drop-down 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 messages. An unsuccessful ping could indicate that the node, gateway, or both are not running.

To ping a remote node:

1. In the Ping a Node to Determine Availability section, select the node in the Message Node Name drop-down 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 8.45 PeopleBook: Integration Testing Utilities and Tools*, “Using the Send Master Utility”.

See *Enterprise PeopleTools 8.45 PeopleBook: Integration Testing Utilities and Tools*, “Using the Simple Post Utility”.

Monitoring Channel Status

Select PeopleTools, Integration Broker, Monitor, Monitor Message, Channel Status to access the Channel Status page.

The Channel Status page displays the status of the channels defined in the local node. A channel has two states: Running and Paused. If you notice a backup within a particular channel, check this page to make sure that the channel is running.

Occasionally, you might need to shut down a particular node to perform a specific maintenance task. Typically, a node contains numerous channels. If you shut down one channel, channels on the node continue processing messages undisturbed.

To determine the status of a channel, locate it within the Channel Name list, and then check the corresponding value in the Status column.

Viewing Channel Partitioning Information

If channel partitioning exists for a channel, a Sub Channel column appears in the results grid on the Message Instances page, the Pub Contracts page, and the Sub Contracts page of the Monitor Message component.

If you click the subchannel name link, the Sub Channel Message Queue page displays and you can view all messages in the sub channel in the order in which they will be processed. You can also resubmit messages or cancel the submission of messages on this page.

Publication Contracts

Sub Channel Message Queue

Channel Name: QE_SALES_ORDER_ASYNC_CHNL **Sub Channel:** QE_F18_ASYNC

Selected Publication Contract

Publishing Node: QE_LOCAL **Pub ID:** 22
Message: QE_F18_ASYNC **Subscriber Node:** QE_IBTGT

Message Queue
Find | View All First 1-5 of 5 Last

	Select	Publishing Node	Pub ID	Subscriber Node	Message Name	Status	Time Stamp	
1	☐	QE_LOCAL	19	QE_IBTGT	QE_F18_ASYNC	Error	11/12/2003 10:27:50AM	Details
2	☐	QE_LOCAL	20	QE_IBTGT	QE_F18_ASYNC	New	11/12/2003 10:27:40AM	Details
3	☐	QE_LOCAL	21	QE_IBTGT	QE_F18_ASYNC	New	11/12/2003 10:27:42AM	Details
4	☐	QE_LOCAL	22	QE_IBTGT	QE_F18_ASYNC	New	11/12/2003 10:31:01AM	Details
5	☐	QE_LOCAL	23	QE_IBTGT	QE_F18_ASYNC	New	11/12/2003 10:31:03AM	Details

☒ Select All ☐ Clear All

[Return to Pub Contracts](#)

Sub Channel Message Queue page

Working with Pub/Sub Server Domains

Select PeopleTools, Integration Broker, Monitor, Monitor Message, Domain Status to access the Domain Status page.

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 messaging 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.

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.

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 Domain Criteria section enables you to perform actions on all domains in the messaging 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 Domain Status section provides application server name and path information for all machines that have domains on the messaging system. For any machine, you can use the drop-down list 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 <i>Inactive</i> .
Refresh	Click to refresh the Domains section and Dispatcher Status section of the page.

Viewing Dispatcher Status

The Dispatcher Status section displays information about machines in the messaging 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, Monitor, Monitor Message.
2. Select the Domain Status tab.

The Domain Status page appears.

3. In the Domains section:
 - a. Locate the row that lists the machine where the domain resides that you want to activate.
 - b. In the Domain Status drop-down list box, select *Active*.
4. Click the Update button.

Inactivating Pub/Sub Server Domains

To inactivate pub/sub servers on domains:

1. Inactivate pub/sub server domains:
 - a. To inactivate domains on all machines in the messaging system, select the All Domains Inactive check box. To activate the servers at a later time, select the All Domains Active box.
 - b. To inactivate domains on individual machines, locate the domains to inactivate. In the drop-down list box, select *Inactivate*. To activate the servers at a later time, select *Activate* in the list.
2. Click the Update button.

The domain status for the domains that you inactivate changes from *Active* to *Inactive*. In addition, in the Dispatcher Status section, the dispatcher status of all processes associated with the domains changes from active (*ACT*) to cleanup (*CLNUP*). Click the Refresh button until the dispatcher status changes to inactive (*INACT*).

If you inactivated all domains, a Force Reset button appears under the Update button. The Force Reset button enables you to force the dispatcher status to change from cleanup to inactive.

Changing Dispatcher Status for Processes

The Force Reset button appears only when you change the domain status for all domains on all machines by selecting the All Domains Inactive check box.

To change dispatcher status for all processes on all machines from cleanup to inactive:

1. Click the Force Reset button.
2. Click Update.

Setting Domain Grace Periods

The time that appears in the Grace Period column indicates when the cleanup process ends. The time equals the system time and the cleanup time interval that you enter.

To set one grace period to apply to domains on all machines, locate the Grace Period for all Domains field in the Domain Criteria section and enter the number of minutes for the grace period. Click Update.

To set grace periods for individual domains, enter the number of minutes for the grace period for each domain. Click Update.

A grace period that you set for an individual domain takes precedence over the setting for all groups.

The grace period setting for all domains is a convenient way to set a grace period for all dispatchers in all the domains. You can set a grace period of all domains at the top of the page and then press the TAB key to access individual domains and override the group setting.

Setting Up Domain Failover

This section discusses domain failover and configuration of domain failover.

To set up domain failover, you use the Failover Configuration page in the Integration Broker Monitor. To access the page, select PeopleTools, Integration Broker, Monitor, Monitor Message, Domain Status. Then, click the Set Up Failover link.

Failover Configuration

☒ **Enable Failover**

IB Failover Time (minutes):

Domains View All First 1-5 of 14 Last			
Failover Group	Failover Priority	Machine Name	Application Server Path
1	1	HLAM032002	C:\pt842\APPSEV\QEMST842
1	2	HLAM042503	C:\pt844\APPSEV\QEMST844
1	3	HLAM072500	C:\pt840\APPSEV\qe84mst
2	1	HLAMCMPQ	C:\pt840\APPSEV\QEMST84
2	2	MDAWSON090803	C:\PT844901B\appser\PT844901B

EMAIL_TO

EMAIL_TO:

EMAIL_CC:

Separate multiple e_mail addresses by semicolon.

Failover Configuration page

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, message 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 messages at that time, the handlers attempt to reboot. If initialization fails, the handlers are not rebooted. If you are using failover, the failover process takes over and failover of the domain occurs.

Domain Failover Groups

You can set up domain failover groups, so that all failover takes place on specific domains. To set up failover groups, you assign a domain a failover group number. After you assign domains to a group, you then assign the failover priority for all domains within the group.

Note. Channel sets within groups should be identical; channel sets between groups should be unique.

The preceding example of the Failover Configuration page 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, *MDAWSON090803*, is the second back-up domain as indicated by the failover priority value 2.

If you do not use dedicated messaging servers, you typically do not need to use groups.

Setting Up Failover on Domains

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) In the Failover Group field, enter a numeric value to specify a group to which a domain belongs.
A value of 1 indicates that the domain is the first backup domain; a value of 2 indicates that the domain is the second back-up domain if the first backup domain fails; and so on.
5. 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.
6. (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.
7. (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.
8. Click the Save button.

Modifying 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.

Viewing Messaging System Performance Statistics

This section provides an overview of messaging system performance statistics and discusses how to:

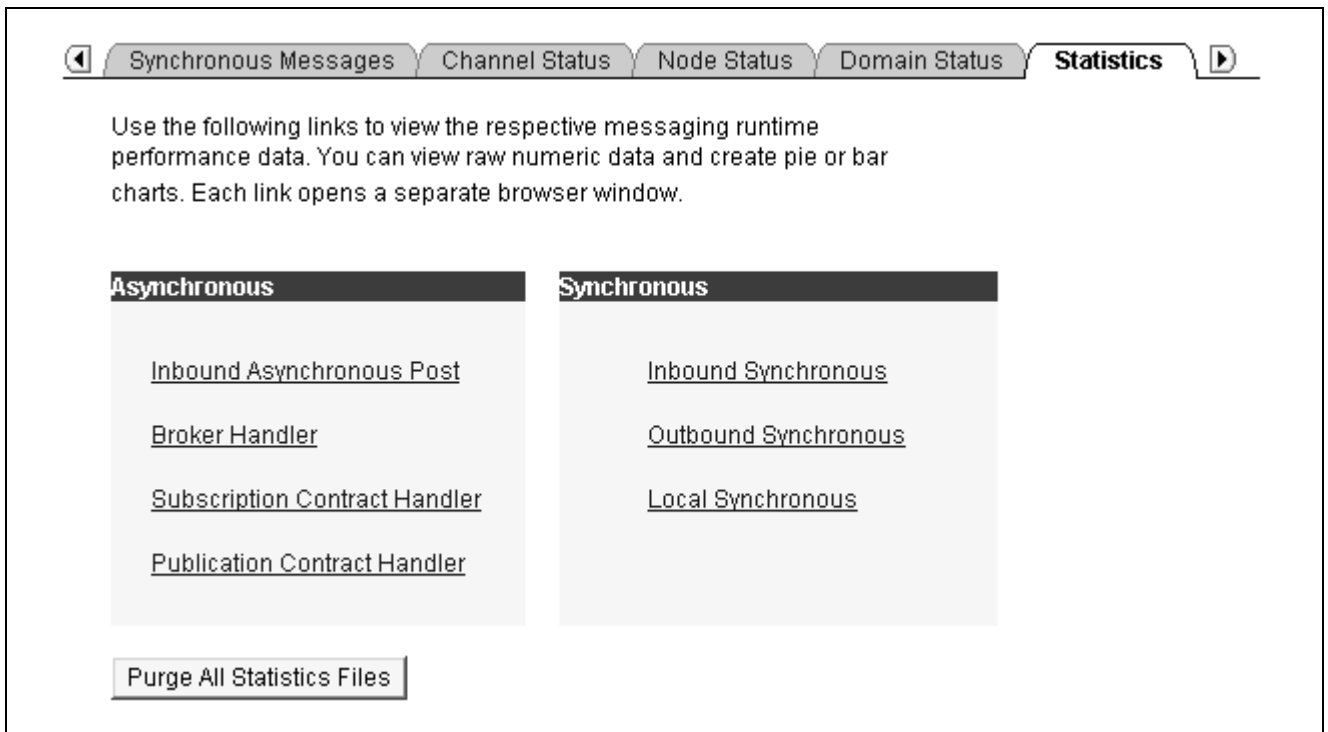
- Enable the messaging system performance statistics feature.
- Filter statistics data.
- View inbound asynchronous post statistics.
- View broker handler statistics.

- View subscription contract handler statistics.
- View publication contract handler statistics.
- View inbound synchronous message statistics.
- View outbound synchronous message statistics.
- View local synchronous message statistics.

Understanding Messaging System Performance Statistics

Integration Broker Monitor provides a Statistics page that enables you to view performance statistics for asynchronous and synchronous messages that flow through the messaging system.

To access the Statistics page, select PeopleTools, Integration Broker, Monitor, Monitor Message, Statistics.



Statistics page in Integration Broker Monitor

Integration Broker Monitor provides performance statistics for the flow of asynchronous messages through the pub/sub system. Statistics are captured for the flow of messages through the following pub/sub components:

Asynchronous Pub/Sub Component	Description
Inbound Asynchronous Post.	Includes posting messages to the integration gateway, as well as posting messages to the application server and putting them into queue. Also includes processing time for creating and sending response messages.
Broker Handler.	Analyzes all messages in the queue and determines the transaction type. Based on the transaction type, creates a subscription contract, publication contract, or both.

Asynchronous Pub/Sub Component	Description
Subscription Contract Handler.	Runs PeopleCode associated with message.
Publication Contract Handler.	Routes the message to another destination.

In addition, Integration Broker Monitor provides performance statistics for the following synchronous transaction types.

Synchronous Transaction Type	Description
Inbound Synchronous.	Inbound messages from PeopleSoft and non-PeopleSoft systems. The sending system requires a response from the receiving PeopleSoft system before additional processing occurs.
Outbound Synchronous.	Outbound messages 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 messages 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 message flow through the publication contract handler and outbound synchronous transactions, you can capture statistics about performance on remote PeopleSoft systems. You do not need to perform any setup tasks in the `integrationGateway.properties` file or in the PSADMIN utility 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.

You can view messaging system performance statistics in numeric or graphical format.

By default, the statistics appear in numeric format on tabbed pages.

Integration Broker Monitor Statistics

Outbound Synchronous

Message Search Criteria

Publish Node
Last

Message Name

Synchronous Timing Data
Customize | Find |
First 1 of 6 Last

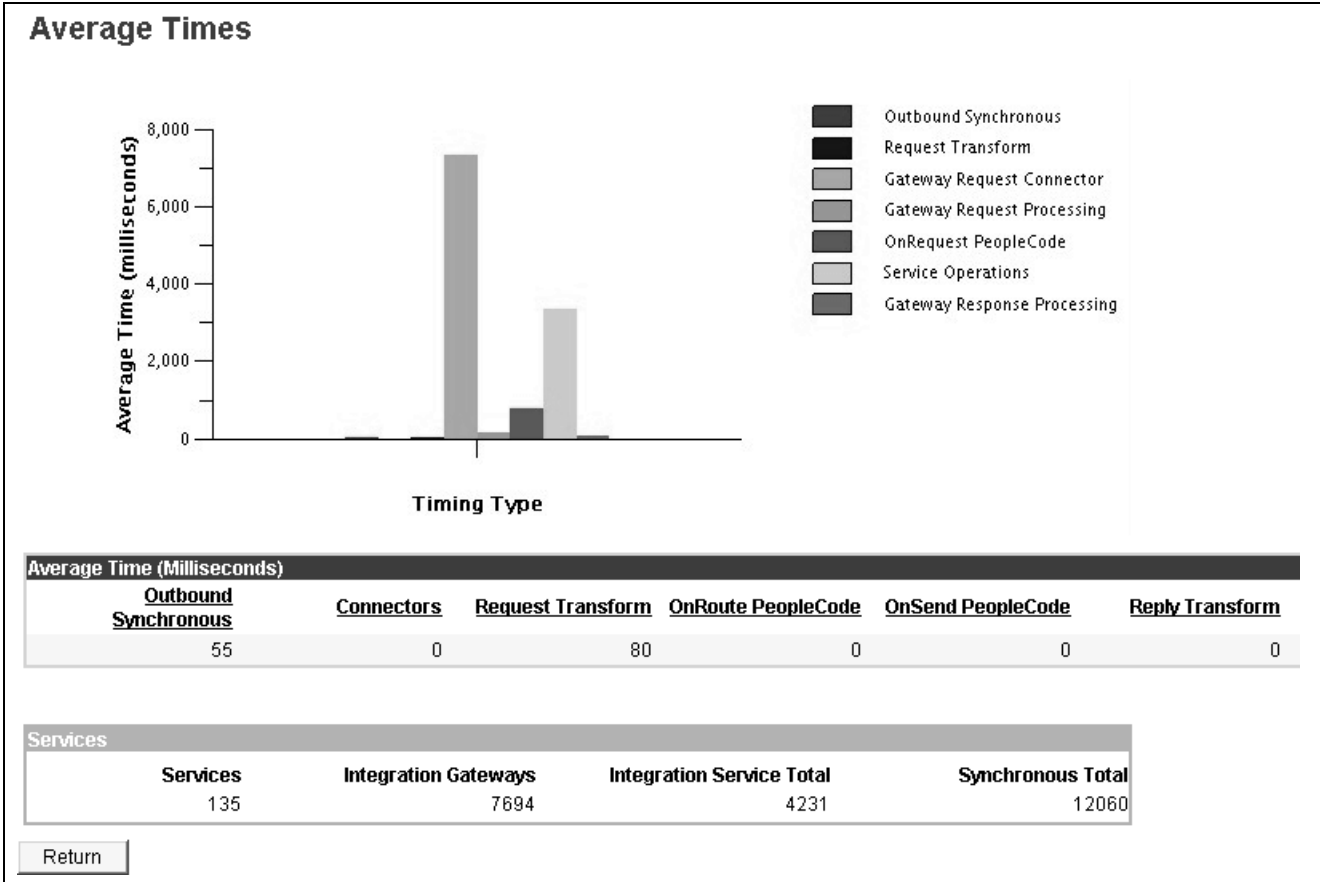
Transactions		Synchronous Total	Synchronous Processing	Gateway Request Times	
Select	Unique Identifier	Message Name	Publishing Node	Timestamp	
1 ☒	37e55a38-8fd7-11d8-b98b-871739837230	QE_FLIGHTPLAN_SYNC	QE_LOCAL	04/16/2004 11:52:36.670000AM	
2 ☒	3d7884c0-8fd7-11d8-b98b-871739837230	QE_FLIGHTPLAN_UNSTRUCT_SYNC	QE_LOCAL	04/16/2004 11:52:47.513000AM	
3 ☒	44574394-8fd7-11d8-b98b-871739837230	QE_FLIGHTPLAN_SYNC_COMBO	QE_LOCAL	04/16/2004 11:52:58.860000AM	
4 ☒	4b63d990-8fd7-11d8-b98b-871739837230	QE_FLIGHTSYNC_TRANSFORM	QE_LOCAL	04/16/2004 11:53:12.240000AM	
5 ☒	877320b4-8fd5-11d8-b98b-871739837230	QE_FLIGHTPLAN_SYNC	QE_LOCAL	04/16/2004 11:40:43.690000AM	
6 ☒	c63bedec-8fd7-11d8-b98b-871739837230	QE_SYNC_FTR_MSG	QE_LOCAL	04/16/2004 11:56:40.340000AM	

Outbound 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.

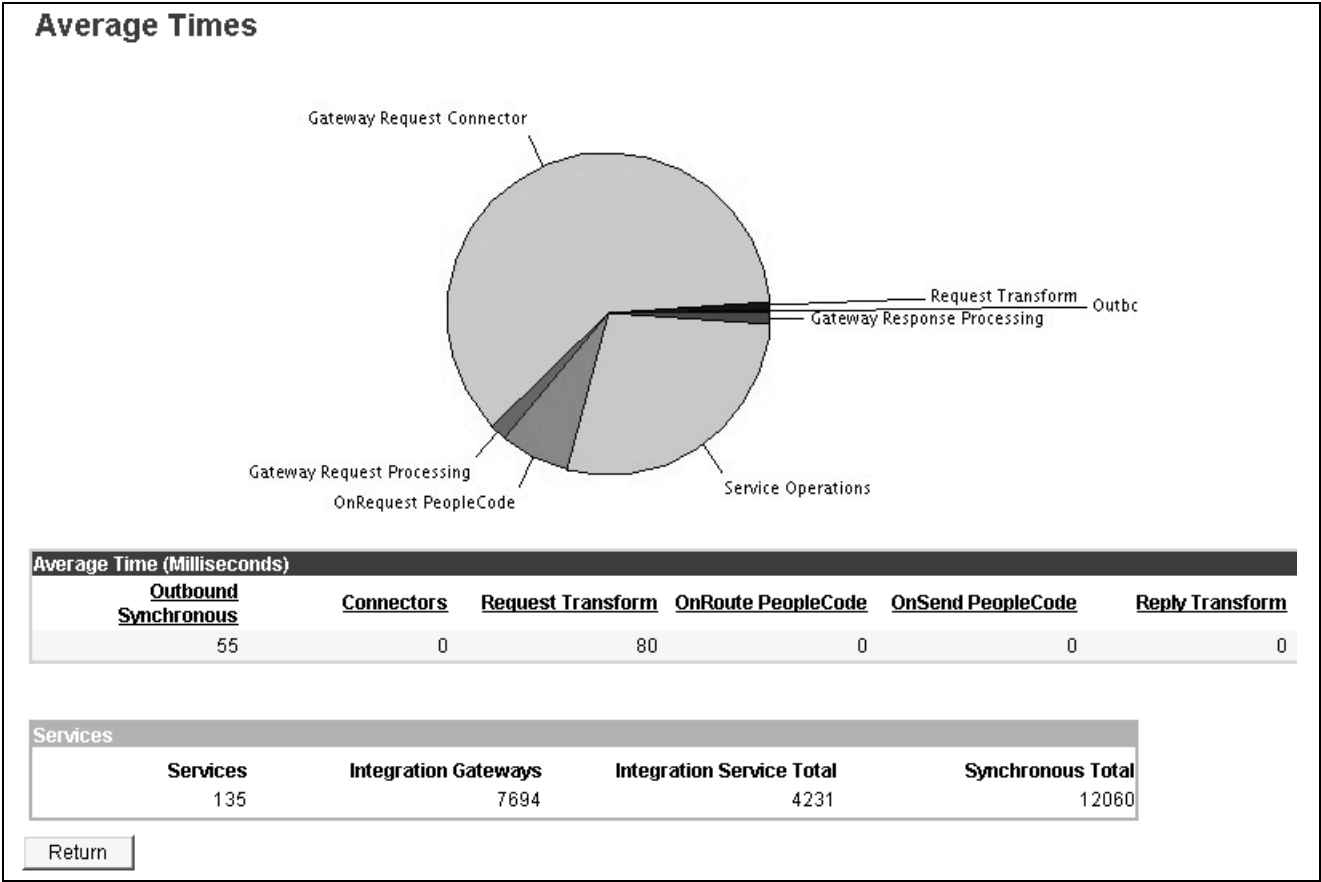
Note. Data displayed in bar chart and pie chart formats are averages.

The following example shows the outbound synchronous performance statistics in bar chart format:



Outbound synchronous performance statistics in bar chart format

The following example shows the same statistics in pie chart format:



Outbound synchronous performance statistics in pie chart format

Common Elements Used in This Section

- Purge All Statistics Files**

Click to purge all messaging system statistics.
- Deselect All**

Click to deselect all nodes.
- 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.
- 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 messaging type when you are done using the statistics or want to view new performance statistics.
- 

Click to remove section tabs and show all columns of information.
- Select**

Check the box to include statistics for the node when viewing the data in bar chart or pie chart format.
- Publishing Node**

Indicates the name of the publishing node.

Unique Identifier	Indicates the unique transaction ID.
Channel, Channel Name	Indicates the name of the message channel.
Publication Identifier	Displays the unique identifier for asynchronous messages.
Message Name	Indicates the name of the message.
Subscription Name	Displays the identifier for PeopleCode that is associated with the message.
Subscriber Node	Displays the name of the subscribing node.
Timestamp	Displays the date and time that the message flowed through the messaging system.

Note. All times shown on the Statistics page are in milliseconds.

Enabling the Messaging System Performance Statistics Feature

To view message system performance information, you enable the statistics feature on the application server through the PSADMIN utility and on the integration gateway through the integrationGateway.properties file. If you do not enable the statistics feature on the application server, the system will not capture any messaging system information.

If you enable the feature only on the application server 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.

To enable the statistics feature on the application server:

1. Access the PSADMIN utility.
2. Locate the PeopleSoft Domain Administration section and select the following option:

Do you want to configure this domain?

3. Enter Y for yes to shut down the domain.
4. At the Do you want to change any config values? prompt, enter Y.
5. Press the ENTER key to move through the list of options, and accept all existing values until you locate the following section:

Values for Config Section - Integration Broker.

6. Change the IB Profile Information setting to Y.
7. Press the ENTER key to finish moving through the lists of options.
8. At the end of the options lists, PSADMIN loads the new configuration.
9. Start up the domain.

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 a transaction type for which to view statistics, you can filter data by various criteria.

Integration Broker Monitor Statistics

Inbound Asynchronous Post

Message Search Criteria

Publish Node: Last:

Channel Name:

Message Name:

Inbound Asynchronous Post Timing Data [Customize](#) | [Find](#) | [View All](#) | First Last

Transactions | Gateway Request Times | Application Server Times |

<u>Publishing Node</u>	<u>Channel Name</u>	<u>Publication Identifier</u>	<u>Message Name</u>	<u>Timestamp</u>
1 ?<no data>				

Message Search Criteria dialog box for inbound asynchronous transactions

The following common elements are used to filter statistics data.

Publish Node	Name of the publishing node. The local default node is the default.
Channel Name	Displays when working with asynchronous messages only. Name of the channel to which a message or messages belong.
Message Name	Name of the message
Last	Use this field to narrow or broaden data display based on intervals of time. Enter a numerical value, and select Days, Hours, or Minutes from the associated drop-down list. Select None to view all messages.
Refresh	Click the button to display results based on the criteria you select or enter.

To view select statistics data to view:

1. Select PeopleTools, Integration Broker, Monitor, Monitor Message and click the Statistics tab. 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 Message Search Criteria dialog box for the transaction type you selected displays.

3. Enter values in the Message Search Criteria dialog box, based on the data you want to view.
4. Click the Refresh button.

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.
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 message. • 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.

Tab	Field	Description and Comments
Application Server Times	Database Insert	Indicates the amount of time to perform the database insert.
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. • Database Insert.

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.
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

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.
Times	Subscription PeopleCode	Indicates the amount of time the systems took to process Subscription 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 Subscription 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.

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	OnAckReceive PeopleCode	Indicates the amount of time that the OnAckReceive PeopleCode event took to run.
Gateway	Gateway Request Transform	Indicates the amount of time to process the request transform.

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	Remote Database	Indicates the amount of time to insert data in the remote database.

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.

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. • Remote Database.
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 Message Statistics

To view inbound synchronous message 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 to View Messaging System Performance Statistics earlier in this section.
Gateway Request Times	Gateway Request Transform	Indicates the amount of time that the system processed the gateway request transform.

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 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.

Viewing Outbound Synchronous Message Statistics

To view outbound synchronous message 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.
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 message.
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.
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.

Tab	Field	Description and Comments
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.

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.

Field	Description and Comments
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 Message Statistics

To view local synchronous message statistics, click the Local Synchronous link on the Statistics page. To view outbound synchronous message 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.
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 message.
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.

Tab	Field	Description and Comments
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.
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.

Running Message Queries

Select PeopleTools, Integration Broker, Monitor, Monitor Message, Queries to access the Queries page.

When you have numerous messages, nodes, and channels it can be difficult to track all of the definitions and routings. Use this page to gather information about messaging system metadata without having to access message definitions, channel definitions, or node definitions in Application Designer.

The queries take advantage of the URL interface for PeopleSoft Query and Process Scheduler Report Manager.

Because PeopleSoft Query is involved, anyone running the queries requires the appropriate security permissions to do so.

These queries are not intended to be run every day or even on a regular basis. They are most useful for obtaining auditing information shortly after an implementation or upgrade. PeopleSoft Query enables you to view the information with Microsoft Excel.

The following table describes each query:

Query	Query Name	Description
Request Message Transactions	PT TRANS FOR MSG RQST	Shows transactions that for a specific request message.

Query	Query Name	Description
Response Message Transactions	PT TRANS FOR MSG RESP	Shows transactions for a specific response message.
Message Status at Node	PT MSGSTATUS AT NODE	Shows the active or inactive messages in the local node.
Messages that belong to a Channel	PT MSGS IN A CHANNEL	Shows the messages that belong to each channel definition.
Channels and Messages at a Node	PT CHNLS MSGS AT A NODE	Shows message status for a node as well as the messages are associated with each channels.
Channels to Archive/Purge Messages	PT CHNL ARCH PURGE AT A NODE	Shows which channels should have their messages archived and which channels should have their messages purged. You can configure the query to prompt for criteria.

See Also

Enterprise PeopleTools 8.45 PeopleBook: PeopleSoft Query

Enterprise PeopleTools 8.45 PeopleBook: PeopleSoft Process Scheduler

Archiving Message Instances

This section discusses archiving message instances and retrieving archived message instances

Archiving Messages

You can archive message instances one at a time from the Message Details component or the Synchronous Details component by clicking the Archive button that appears in the Action column on either page.

Retrieving Archived Messages

You can retrieve archived message instances from the following tabs in the Monitor Message component: Overview tab, Message Instances tab, Pub Contracts tab, and Sub Contracts tab.

To retrieve archived message instances, select the Archive check box and click Refresh. Archived messages appear in the results grid. For any returned row, click the Details link to view the message header and message content.

Monitoring Asynchronous Message Details

The Message Details component enables you to gather in-depth information about a specific asynchronous message. It also enables you to perform tasks such as correct errors and resubmit messages.

This section discusses how to:

- Access the Message Details component.
- View asynchronous message properties.
- View asynchronous message errors.
- View asynchronous messages in XML format.
- View nonrepudiation information for asynchronous messages.

Accessing the Message Details Component

To access the Message Details component, use the PeopleTools menu structure and select PeopleTools, Integration Broker, Monitor, Message Details. The Message Details search page displays.

You can search for a message using the following criteria:

Channel Name	Displays a list of messages assigned to a channel you specify.
Last Update Date/Time	Displays a list of messages based on the date and time they were updated. The format is mm/dd/yyyy 00:00:00.
Publication Identifier	Displays a list of messages based on publication ID in descending order, so that the most recently published messages display first.
Publishing Node	Displays a list of messages based on a publishing node you specify.

After you search for and select a message, the Message Detail —Message properties page appears.

The screenshot displays the 'Message Details' page with three main sections:

- Message Instance Information:** A table showing details for a specific message instance.

Message Instance Information		Actions
Pub Node	QE_LOCAL	Resubmit Cancel Archive View XML View IB Info
Channel	IB_CHNL	
Pub ID	6	
Message	QE_IB_REL_OAOA_MSG	
Version	VERSION_1	
Trans Type	OA	
Process	QE_IB_MSG_TEST	
Time Stamp	03/10/2004 12:41:26PM	
Publisher	QEDMO	
Status	Done	
NRID		
Data Length	2780	
Data Length View Limit	100000	
- Publication Contracts:** A table showing the contract for the message.

Subscriber Node	Status	Actions
QE_IBTGT	Done	Resubmit Cancel View XML View IB Info
- Subscription Contracts:** A table showing the subscription for the message.

Subscription	Status

Message Details—Message Properties page

Viewing Asynchronous Message Properties

Select PeopleTools, Integration Broker, Monitor, Message Details to access the Message Properties page.

The Message Properties page displays information pertaining to the message instance and all associated publication and subscription contracts.

There are three sections that appear on the Message Properties page. This section discusses the three grids and the information that appears in them. This section discusses:

- Message Instance Information section.
- Publication Contracts section.
- Subscription Contracts section.

Message Instance Information Section

The Messaging Instance Information section provides general information pertaining to a particular message to assist in troubleshooting.

Pub Node (publication node)	Identifies the node that published the message.
Channel	Identifies the channel to which the message is associated.
Pub ID (publication ID)	Identifies the Publication ID. Unique identifier for message. Applies to asynchronous messages only.
Message	Identifies the message name.
Dflt data ver (default data version)	Identifies the default data version.
Trans Type (transaction type)	Identifies the transaction type. Values are: • <i>OA</i>: Outbound asynchronous. • <i>IA</i>: Inbound asynchronous.
Pub Process (publication process)	Identifies the process that published the message; the name of the component that published the message.
Time Stamp	Identifies the date and time that the message instance was last processed.
Publisher	Identifies the publisher of the message. The publisher is usually the user ID of the person in the publishing system who triggered the message publication.
Status	Identifies the status of the message, such as <i>Done</i> , <i>Error</i> , <i>Started</i> , and so on.
NRID (nonrepudiation ID)	Identifies a nonrepudiation ID. This ID is a unique number used to associate a message instance with the nonrepudiation log.

The Actions column in the Message Instance section contains controls that apply to the message instance and all associated publication and subscription contracts.

Resubmit Click the Resubmit button to resubmit a message for processing. This button is enabled when a message has a status of *Time Out*, *Error*, *Edited*, or *Cancelled*. If a message contains an error or has timed out, typically you can just correct the problem and resubmit the message. After you edit a message, the status becomes *Edited*. When you resubmit the message, the status changes, yet again, to *New*.

Cancel	Click the Cancel button to cancel processing attempts for a message. This button is enabled with a message has a status of <i>New</i> , <i>Retry</i> , <i>Time Out</i> , <i>Error</i> , or <i>Edited</i> .
Archive	Click the Archive button to archive a message. This button is enabled when a message has a status of <i>Done</i> or <i>Cancelled</i> .
Data Length	Indicates the size of the XML message in bytes.
Data Length View Limit	Indicates the maximum size of messages in bytes that are automatically loaded in the XML Viewer. The default is 100000 bytes. You can change the default in the psappsrv.cfg file using the Message View Size property in the [Integration Broker] section of the file. Do not set this value to a negative number.
View XML	The View XML link appears when the size of the message, as indicated in the Data Length field, is less than the message size limit specified in the Data Length View Limit field. Click the link to view the XML data that was received for the selected message instance.
Download XML	The Download XML link displays when the size of the message, as indicated in the Data Length field, exceeds the message size limit, as specified in the Data Length View Limit field. Click the link to download the XML data that was received for the selected message instance to a location you specify.
Upload XML	The Upload XML link displays when the size of the message, as indicated in the Data Length field, exceeds the message size limit specified in the Data Length View Limit field. This feature complements the download XML feature, and enables you to upload previously downloaded XML data back into the system. When you click the link, a dialog box displays that enables you to specify the location of the file to upload.
View IBInfo	Click the link to view IBInfo data for a message, such as requesting node, transaction ID, content type and so on.

Publication Contracts Section

The Actions tab reveals all the nodes subscribing to a particular message and the current status of the publication contract, as in whether the publication has been successfully posted to the subscribing node.

If you click the View XML link, a new page opens and the XML for the selected message appears. If any transformations were applied for the publication contract, the transformed XML displays. Use the View XML link in the Message Instance section to see the original XML that was received. You can edit the XML if you have the appropriate permissions, and if the message has the status of *New*, *Error*, *Retry*, *Timeout*, *Edited* or *Cancelled*.

Click the View IBInfo link to view IBInfo data for the publication contract, such as requesting node, transaction ID, content type, and so on.

See [Chapter 14, “Using Integration Broker Monitor,” Message Statuses, page 301](#).

The Information tab contains the following information about the publication contract:

Subscriber Node	Identifies the name of the subscribing node.
Status	Identifies the processing status.
Time Stamp	Identifies the time that the system last modified the publication contract.
Retry Count	If the first attempt to deliver the message failed, this value reflects the number of times the system has attempted to redeliver the message.
Machine Name	Identifies the name of the local application server machine that processed the publication contract.
PID (process ID)	Identifies the process ID on the local application server. It shows the PID of the PSMSGHND (handler) that created the contract.
NRID	Identifies the nonrepudiation ID. The NRID is a unique number used to associate a message instance with the nonrepudiation log.
Trans Type (transaction type)	Identifies the transaction type. Values are: • <i>OA</i>: Outbound asynchronous. • <i>IA</i>: Inbound asynchronous.
Request Message Name	Identifies the request message name. This message name can be different than the name in the message instance due to a relationship.
App. Msg Ver (application message version)	Identifies the message version.

Subscription Contracts Section

The Actions tab reveals the status of a particular subscription contract.

If you click the View XML link, a new page opens and the XML for the selected message displays. If any transformations were applied for the Subscription Contract, the transformed XML displays. Use the View XML link in the Message Instance section to see the original XML that was received.

You can edit the XML if you have the appropriate permissions, and if the message has the status of *New*, *Error*, *Retry*, *Timeout*, *Edited* or *Cancelled*.

See [Chapter 14, “Using Integration Broker Monitor,” Message Statuses, page 301](#).

The Information tab provides the following information.

Subscription	Identifies the subscription process name.
Status	Identifies the processing status.
Time Stamp	Identifies the time that the system last modified the subscription contract.
Retry Count	If the first attempt to subscribe to the message failed, this value reflects the number of times the system has attempted to subscribe to the message.
PID (process ID)	Identifies the process ID on the server.
SubProcID (subscription process ID)	Identifies the subscription process ID associated with the subscription contract for identification purposes.
Trans Type (transaction type)	Identifies the transaction type. Values are:

- *OA*: Outbound asynchronous.
- *IA*: Inbound asynchronous.

App. Msg Ver(application message version) Identifies the message version.

Viewing Asynchronous Message Errors

The Message Errors page enables you to view any possible errors with message instances, publication contracts, and subscription contracts.

The tabs on this page let you see when an error occurred, what the error message is, and where the error occurred.

Viewing Asynchronous Messages in XML Format

The XML Message Viewer page in the Message Details component enables you to view the raw XML code. On this tab, you can locate specific sections of the XML code.

Use the Display Formatted check box to adjust how the XML appears. In most cases, displaying the message in its formatted form makes it easier to read and to locate specific values or tags.

The XML Message Viewer page also features several links.

Display Formatted	Select to adjust how the XML appears. In most cases, displaying the message in its formatted form makes it easier to read and to locate specific values or tags.
Edit XML	Click the Edit XML link to edit the XML displayed. The button is enabled provided you have the appropriate permissions and the status of the message is <i>New</i> , <i>Error</i> , <i>Retry</i> , <i>Timeout</i> , <i>Edited</i> or <i>Cancelled</i> .
Print XML	Click the Print XML link to print the XML. When you click the button, the XML appears in a new window. Use the Print command on your browser to print the page.

Viewing Nonrepudiation Information for Asynchronous Messages

The Signature (NR) page of the Message Details component displays nonrepudiation information for asynchronous messages. This page appears only if a message is sent with a signature; it displays the nonrepudiation signature in XML format.

Click the Confirm button to confirm the nonrepudiation status.

Viewing Synchronous Message Details

The Synchronous Details component enables you to gather in-depth information about a specific synchronous message. It also enables you to perform tasks such as view logs and correct errors.

Accessing the Synchronous Details Component

To access the Synchronous Details component, select PeopleTools, Integration Broker, Monitor, Synchronous Details. By default, the Sync Message Detail page is active.

Sync Message DetailSync Message Errors

Orig Pub Node:QE_LOCALChannel:QE_SALES_ORDER_SYNC_CHNL

GUID:2131ba76-1dd2-11b2-9441-d5cb7b682244

Message:QE_SALES_ORDER_SYNCApp. Msg Ver:VERSION_1

Trans Type:InSyncStatus String:DONE

Archive

Message Name Detail:Publisher:VP1

Non-Repudiation ID:4Signature

Publishing Node:QE_LOCAL

Destination Publish Node:QE_LOCALFinal Destination Node:

Pub/Sub Timestamp:01/24/02 5:01:31.000000PMLast Upd DtTm:01/24/02 5:01:31PM

Log Type:Request - OriginalView XML

PeopleTools, Integration Broker, Monitor, Synchronous Details

The following sections provide information about each page in the Synchronous Details component.

Viewing Synchronous Message Details

Select PeopleTools, Integration Broker, Monitor, Synchronous Details to access the Sync Message Detail page.

The Sync Message Detail page provides read-only information about synchronous messages in the system. It also enables you to view signature information for a message if it was processed with nonrepudiation logic.

In addition, you can use the Archive button to archive messages.

The section discusses:

- Viewing message attributes.
- Viewing message signature information.

Viewing Message Attributes

You can view the following message attribute information on the Sync Message Detail page.

Note. For synchronous messages, you can view the full message detail in XML only if you have selected the Log Message Detail option for the transactions. To select this option, select Node Definitions, Transactions, Edit (a transaction), Transaction Detail, Message.

Orig Pub Node (original publishing node)	Identifies the node that published the message.
Channel	Identifies the channel to which the message is associated.

GUID (global unique identifier)	Identifies the Global Unique Identifier (GUID) that uniquely identifies each message. GUID applies to only to synchronous messages.
Message	Identifies the message name.
Version	Identifies the message version.
Trans Type (transaction type)	Identifies the transaction type. Values are: • <i>OutSync</i>: Outbound Synchronous • <i>InSync</i>: Inbound Synchronous
Status	Identifies the status of the message, such as Done, Error, Started, and so on.
Message Name Detail	Reserved for future use.
Publisher	Publisher of the message. This is usually the user ID of the person in the publishing system who triggered the message publication.
Time Stamp	Identifies the date and time that the message instance was last processed.
Publisher	Identifies the publisher of the message. This is usually the user ID of the person in the publishing system who triggered the message publication.
Non-Repudiation ID	Identifies a unique number used to associate a message instance with the nonrepudiation log.
Signature	Click to view the nonrepudiation signature for the message in XML format.
Publishing Node	Identifies the node that published the message.
Destination Publish Node	Identifies the name of the node where the message will be sent.
Pub/Sub Timestamp	Indicates the time and date of when the message was published.
Last Upd DtTm (last update date/time)	Identifies the date and time the message was last updated.
Log Type	Select a value from the drop-down list box and click the View XML link to view the corresponding information. Values are: • <i>Request Original</i>: Displays the original request data in XML format. • <i>Request Transformed</i>: Displays transformed request data, if applicable, in XML format. • <i>Response Original</i>: Displays the original response data in XML format. • <i>Response Transformed</i>: Displays the transformed response data, if applicable, in XML format.
View XML	Click to view the message content in XML format.

Viewing Message XML Content

In certain situations, the XML content of a message isn't visible in Integration Broker Monitor. This is because of the way message data is logged. Initially, the log data (including the message XML) for any transaction is held in system memory.

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 message XML content right after the message was transmitted, exit the Synchronous Details component and wait for a minute, then reopen the message and check the XML view again.

Note. The synchronous message XML content won't appear at all unless you selected Log Message Detail on the Transaction Messages page of the synchronous transaction definition.

See [Chapter 9, “Configuring Nodes and Transactions,” Configuring Transactions, page 204](#).

Viewing Message Signature Information

If a message is sent with a signature, a signature link appears next to the Non-Repudiation ID number. When you click the signature link, the message signature appears in XML. You can click the Confirm button to confirm the nonrepudiation status. Click the Return button to return to the Sync Message Details page.

Viewing Synchronous Message Errors

The Sync Error page in the Synchronous Details component provides read-only information about any error related to synchronous messages.

The information that appears on the page indicates the original publishing node of the message, the channel, the message name, the transaction type, and more.

Note. If you have the Synchronous Logging option set to *No Logging* for a synchronous message transaction, the message header and error information is displayed only if there is an error. If there is no error, no information is logged and displayed.

The tabs on this page let you view error message details such as what type of error occurred, and when and where the error occurred.

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 Integration Broker Monitor to scan your system for messages, that approach requires you to launch Integration Broker 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, Error Notification.

Note. You can use PT_AMM_WF to notify users of errors relating to asynchronous messages only.

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 messages with a status of either Error or Timeout. • Publications Queue Publications Contracts Queue • Subscriptions Contracts Queue
2	Trigger Workflow	Upon encountering a message 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 message with a user through the message's channel name property. All users that have at least read-access to the message channel are notified.
3	Resolve Issue	Administrators also receive a new worklist item reflecting the problematic message. To access the message, an administrator clicks the item in the worklist. The link leads to the Message Details component. The component is presented with the specified message 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 and 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 8.45 PeopleBook: Security Administration

Creating Static Error Notification Lists

By default, error notifications are sent to all users who can monitor the message channel; 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.

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.
 - b. Select Roles.
 - c. Select and open the APP_MSG_ADMINISTRATOR role.
 - d. Click the Workflow tab.
 - e. 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 8.45 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, Error Notification.

Error Notification

Run Control ID: Test [Report Manager](#) [Process Monitor](#) [Run](#)

Process Frequency

☐ Process Once *Request ID:
☐ Always Process Description:
☒ Don't Run URL:

Error Notification page

To run PT_AMM_WF:

1. Select PeopleTools, Integration Broker, Monitor, 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.

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.

Running Batch Message Archiving Processes

For performance and general maintenance reasons, you may want to archive older messages to clear space on your live messaging tables.

The PeopleSoft system provides an Application Engine program that scans all of the messaging tables in the system for message archiving purposes. You use the Archive Messages component (RUN_APMMSGARCH) to access the program.

You use the Run Archive page to invoke the archive process. To access the page, select PeopleTools, Integration Broker, Monitor, Archive Messages.

Run Archive

Run Control ID: 2 [Report Manager](#) [Process Monitor](#) [Run](#)

☐ Archive All Messages

Archive Specific Messages

Status: ☐ Done ☐ Cancel

Older Than: days

Run Archive page

You can use the Run Archive page to archive all messages with a status of *Done* or *Cancel*. Or you can archive messages based on their status, their age, or a combination of the two. For example, you can choose to archive messages with a status of *Done* that have been in the messaging system for more than 14 days.

Before you run a batch message archive process, inactivate the pub/sub server domain. Then, after you run the process, reactive the pub/sub server domain.

See [Chapter 14, “Using Integration Broker Monitor,” Working with Pub/Sub Server Domains, page 312.](#)

To run the batch message archiving process:

1. Select PeopleTools, Integration Broker, Monitor, Archive Messages.

2. Select an existing run control ID, or add a new one.

The Run Archive page appears.

3. Select the messages to archive

Archive All Messages	Select this check box to archive all messages in the messaging system with a status of <i>Done</i> or <i>Cancel</i> , regardless of how long they have been in the messaging system.
Done	Select to archive messages in the messaging system with a status of <i>Done</i> .
Cancel	Select to archive only those messages in the messaging system with a status of <i>Cancel</i> .
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 messages based on days. If the date is August 15, 2004, messages 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.

Note. Using APPMSGARCH to archive message data is the batch approach. You can also archive individual messages online using the Integration Broker Monitor.

See Also

Enterprise PeopleTools 8.45 PeopleBook: PeopleSoft Process Scheduler, “Using Process Monitor”

Enterprise PeopleTools 8.45 PeopleBook: PeopleSoft Process Scheduler, “Using Report Manager”

Purging Messaging Tables

The PeopleSoft system provides a collection of Data Mover scripts that you can run to purge the messaging 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.

Script Name	Description
AppMsgPurgeAll.dms	Deletes queue data from every archive or live message 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.

Script Name	Description
AppMsgPurgeArchive.dms	Deletes queue data from every archive message table in the database.
AppMsgPurgeLive.dms	Deletes queue data from every live message table in the database.

Using the Message Monitor Component Interface

Integration Broker Monitor includes a collection of inquiry methods that you can access with a component interface.

Use the MSGSTATUSSUMMARY component interface to extract information from Integration Broker Monitor. The output of the component interface reveals the amount of contracts that are in the queue. The contracts appear grouped by status and message or grouped by status and channel.

You can use the following user-defined methods to extract information:

- FillPubConByMsg()
- FillPubConByChannel()
- FillSubConByMsg()
- FillSubConByChannel()

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, oConnectionString, 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()
    '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 8.45 PeopleBook: PeopleSoft Component Interfaces,
“Introducing Component Interfaces”

CHAPTER 15

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.

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.

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.

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 4, “Managing Integration Gateways,” Setting Logging Properties, page 48](#)

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.

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 <code>IBRequest</code> or <code>IBResponse</code> . If another object is passed, the <code>toString</code> method is invoked for the object, and the result is logged.
MessageLevel	Set the relative importance of the information that you are logging. The <code>ig.log.level</code> property setting in the <code>integrationGateway.properties</code> file determines the log level that is currently in effect. If the <code>MessageLevel</code> value that is passed here is less than or equal to the <code>ig.log.level</code> property setting, the message is written to the log file. Values are: • 3: Important information. • 4: Standard information. • 5: Low-importance information. The <code>ig.messageLog.filename</code> property in the <code>integrationGateway.properties</code> 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.

- 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.

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 8.45 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 8.45 PeopleBook: System and Server Administration*, “Setting Application Server Domain Parameters,” Trace Options.

- Set the level of network tracing (log fence).

See *Enterprise PeopleTools 8.45 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 12, “Applying Filtering, Transformation and Translation,” page 247

Enterprise PeopleTools 8.45 PeopleBook: PeopleSoft Application Engine, “Tracing Application Engine Programs”

Debugging Integrations

This section discusses how to:

- Debug subscription PeopleCode.
- Handle common issues.

Debugging Subscription PeopleCode

To debug Subscription PeopleCode:

1. Start PeopleSoft Application Designer on the server machine that is processing the subscription.
You must use the same user ID and database as those for which the server machine’s application servers are configured.
2. Start debug mode.

When the subscription server processes the subscription PeopleCode, the debugger starts.

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 error log. The default file location is <PS_HOME>\webserv\<DOMAIN> \applications\peoplesoft\PSIGW\errorLog.html.
Integration gateway.	Check the integration gateway message log. The default file location is <PS_HOME>\webserv\<DOMAIN> \applications\peoplesoft\PSIGW\msgLog.html.
Message channels are paused.	Check the Integration Broker Monitor. Select PeopleTools, Integration Broker, Monitor, Monitor Message, Channel Status.
A node is paused.	Check the Integration Broker Monitor. Select PeopleTools, Integration Broker, Monitor, Monitor Message, 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, Gateways. V
Node inactive.	Check the node definition. Select PeopleTools, Integration Broker, Node Definitions.
Subscription PeopleCode is missing or incorrect.	Check the Integration Broker Monitor. Select PeopleTools, Integration Broker, Monitor, Monitor Message, Sub Contracts. Select the Queue Information tab. Also check the Message Designer in PeopleSoft Application Designer.

Area or Suspected Issue	Debugging Suggestion
A message is inactive.	Check the Message Designer in PeopleSoft Application Designer.
There are transform problems.	• Check the Application Engine object in PeopleSoft Application Designer. • For before and after images, check the Message Monitor in Integration Broker Monitor. For asynchronous messages, select Integration Broker, Monitor, Message Details. On the Message Properties tab, click View XML for the publication contract or subscription contract. For synchronous messages, select Integration Broker, Monitor, Synchronous Details. On the Sync Message Detail tab, use the Log Type drop-down 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: <code><PS_HOME>\appserv\<Domain>\psappsrv.cfg</code> 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: <code><PS_HOME>\appserv\<Domain>\LOGS\ <operID>_<machine name>.AET</code> 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.

CHAPTER 16

Creating Third-Party Integrations Using WSDL

PeopleSoft provides you with the ability to create inbound and outbound integrations with third-party systems, including enterprise resource planning (ERP) systems, using Web Services Description Language (WSDL). This chapter discusses how to:

- Perform Integrations using WSDL.
- Use iWay SOAPswitch.
- Start and stop iWay SOAPswitch.
- Log in to iWay SOAPswitch.
- Generate WSDL using ERP connectors.
- Import WSDL into PeopleSoft.
- Use the WSDL repository.
- Create service transaction metadata for imported WSDL.
- Generate WSDL from PeopleSoft component interface and message definitions.

Understanding Integrations Using WSDL

A WSDL document is an XML file that contains information about the interface, semantics and other details of a call to a web service.

When someone wants to use a service, they request a WSDL document to find out the location of the service, the function calls and how to access them. They then use this information in the WSDL to form a Simple Object Access Protocol (SOAP) request and send it via HTTP to the external system endpoint.

PeopleSoft provides the capability to manually import WSDL into PeopleSoft via a file or URL. PeopleSoft also provides access to a third-party software product, iWay SOAPswitch, that provides ERP adaptors, or connectors, that enable you to generate WSDL from SAP, Oracle and Siebel systems, and import it into PeopleSoft. You can then use the Operation Wizard to create transaction metadata. From there you can create any necessary transformations, write required PeopleCode, and execute the integration.

PeopleSoft also enables you to expose PeopleSoft component interface and message definitions as WSDL to create integrations.

To perform integrations using WSDL you should have an understanding of the following:

- WSDL.
- SOAP.
- Creating transformations using Extensible Stylesheet Transformations (XSLT) and PeopleCode.

- Concepts of setting up integration broker metadata, such as setting up nodes, messages, message channels and so on.

Terminology

The following terminology is used when creating integrations using WSDL.

WSDL	According to the World Wide Web Consortium (W3C), “WSDL is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define an endpoint. Related concrete endpoints are combined into abstract endpoints (services). WSDL is extensible to allow description of endpoints and their messages regardless of what message formats or network protocols are used to communicate . . .”
Web service	According to the W3C, “A web service is a software application identified by a URI [RFC 2396], whose interfaces and bindings are capable of being defined, described, and discovered as XML artifacts. A web service supports direct interactions with other software agents using XML based messages exchanged via internet-based protocols. A collection of Endpoints.”
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.
Message	According to the W3C, “an abstract, typed definition of the data being communicated. “
Operations	According to the W3C, “An abstract description of an action supported by the service.”
PortType	According to the W3C, “An abstract set of operations supported by one or more endpoints.”
Binding	According to the W3C, “A concrete protocol and data format specification for a particular port type.”
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.”
Service	According to the W3C, “A collection of related endpoints.”

See Also

<http://www.w3.org/TR/wsdl>

Bindings

PeopleSoft supports WSDL bindings that define document and remote procedure call (RPC) styles.

If you use RPC bindings you must develop the appropriate transformations to perform the RPC calls.

Understanding iWay SOAPswitch

iWay SOAPswitch is a wizard-driven product that enables you to expose software functionality via 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 back-end system responses.

Installing iWay SOAPswitch

iWay SOAPswitch is an installation option during the PeopleSoft Pure Internet Architecture installation process.

During the installation process you are prompted to install and specify a location for ERP connectors. Choosing to install the ERP connectors is synonymous with installing iWay SOAPswitch.

Components

iWay SOAPswitch is comprised of three main components:

SOAPswitch Server	The SOAPswitch server listens for SOAP requests from web service consumers and forwards requests to back-end servers via SOAPswitch adapters.
Administration Console	The Administration Console allows 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/clients.

Delivered Adapters

The following adapters are provided with the iWay SOAPswitch product delivered with PeopleTools 8.45:

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 supports and enables you to access stored procedures, tables, and views for Oracle applications that run on Oracle 8i and 9i databases. The interfaces are made available as web services to be used by any consumer that can leverage WSDL and create SOAP requests for the proxied service. SOAPswitch focuses on

Oracle Financials and Oracle Project as reference implementations for the two primary Oracle application methodologies.

At this time the Oracle adapter only supports sending messages from PeopleSoft to Oracle. The ability for Oracle to send messages to PeopleSoft is not currently supported.

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 messages between PeopleSoft and SAP systems.

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 messages from PeopleSoft to Siebel. Use the XML adapter to send messages from Siebel to PeopleSoft.

XML Adapter

The iWay SOAPswitch XML Adapter enables you to set up and access XML objects for use with SOAPswitch.

Operation Types

An iWay SOAPswitch operation type is equivalent to a PeopleSoft transaction type. The following table maps PeopleSoft transaction types to their equivalent iWay SOAPswitch operation types.

PeopleSoft Transaction Type	Equivalent iWay SOAPswitch Operation Type
Outbound asynchronous transaction.	Notifications operation.
Inbound asynchronous transaction.	One-way operation.
Outbound synchronous transaction.	Solicit-Response operation.
Inbound synchronous transaction.	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 defined on iWay SOAPswitch specifies inbound asynchronous or synchronous transactions to SOAPswitch.

Notifications specify outbound asynchronous or synchronous transactions from iWay SOAPswitch.

MySQL Database

iWay SOAPswitch delivers MySQL as the default database for event and audit logging. However, you can use other databases for this purpose.

See Also

iWay SOAPswitch Administration Guide, “Chapter 3: Configuring Your SOAPswitch Environment,” *Logging*.

Jetty Servlet Server

The iWay SOAPswitch product that ships with PeopleTools uses a stand-alone Jetty servlet server. The Jetty server is bundled with and is automatically installed when you install SOAPswitch. The Jetty servlet server is installed in the \temp directory, not in <SOAPSwitch_HOME>.

Running iWay SOAPswitch as a servlet under BEA WebLogic 8.1 or IBM WebSphere 5.1 is not yet supported.

More information about the Jetty servlet server is located in a readme.txt file located in the \soapswitch\Jetty-4.2.11 directory.

iWay SOAPswitch and Adapter Documentation

This PeopleBook provides summary information about tasks you perform in iWay SOAPswitch and with the iWay SOAPswitch adapters, and provides cross references to iWay SOAPswitch documentation whenever possible. PeopleSoft recommends that you thoroughly review the iWay SOAPswitch documentation to take full advantage of the product.

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 displays.
2. From the left navigation area, select Documentation.

Documentation List

The following iWay SOAPswitch documentation is provided in HTML and PDF formats:

- *SOAPswitch Getting Started Guide*
- *SOAPswitch Administration Guide*
- *Managing Existing Web Services Guide*

The following iWay SOAPswitch adapter documentation is available.

- *J2EE Adapter Guide*
- *Oracle Applications Database (OAP) Adapter Guide*
- *SAP R/3 Adapter Guide*
- *Siebel Adapter Guide*
- *XML Adapter Guide*

Starting and Stopping iWay SOAPswitch

This section describes how to:

- Start SOAPswitch Server.
- Stop SOAPswitch Server.

Understanding Starting and Stopping iWay SOAPswitch

To start and access iWay SOAPswitch, you must start SOAPswitch server.

To stop iWay SOAPswitch, you stop SOAPswitch server.

Common Elements Used in This Section



In Windows, the icon that displays on the system tray when SOAPswitch Server is running.

Starting iWay SOAPswitch Server

To start the SOAPswitch Server:

- In the Windows environment, select Start, Programs, iWay SOAPswitch, Start SOAPswitch Server. When SOAPswitch server is running, the server icon displays on the system tray.
- In the UNIX environment, launch the file `runsoapswitch.sh` located in the `<ERPConnector_Install_Dir>/bin`.

Stopping iWay SOAPswitch Server

This section discusses how to stop SOAPswitch Server.

Stopping SOAPswitch Server in Windows

To stop SOAPswitch Server in the Windows environment:

1. Right-click the SOAPswitch Server icon in the system tray.
2. Select Close.

Stopping SOAPswitch Server in UNIX

To stop SOAPswitch Server in the UNIX environment:

1. Navigate to `<ERPConnector_Install_Dir>/bin`.
2. Enter the following command:

```
runsoapswitch.sh stop
```

Logging In to iWay SOAPswitch

This section discusses how to:

- Log in to iWay SOAPswitch.
- Change the iWay SOAPswitch user ID and password.

Logging In to iWay SOAPswitch

Login information for iWay SOAPswitch is stored in 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.

The screenshot shows the 'ERP Connectors Admin' page. At the top, there is a field for '*ERP Connector API URL:' with the value 'http://localhost:4400/ssw/api'. Below this is a 'Login Information' section with a gray header. Inside this section, there are three input fields: '*User ID:' with the value 'admin', '*Password:' with the value '*****', and 'Confirm Password:'. To the right of the *User ID field is a checkbox labeled 'Change User ID'. To the right of the *Password field is a checkbox labeled 'Change Password'. At the bottom right of the Login Information section are two buttons: 'Save' and 'Cancel'.

ERP Connectors Admin page

This login provides access to the iWay SOAPswitch functionality in the PeopleTools, Integration Broker, Web Services, 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 *4400* is the port number:

`http://<machinename>:4400/ssw/api`

Ports 4400 and 8080 are the default ports 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

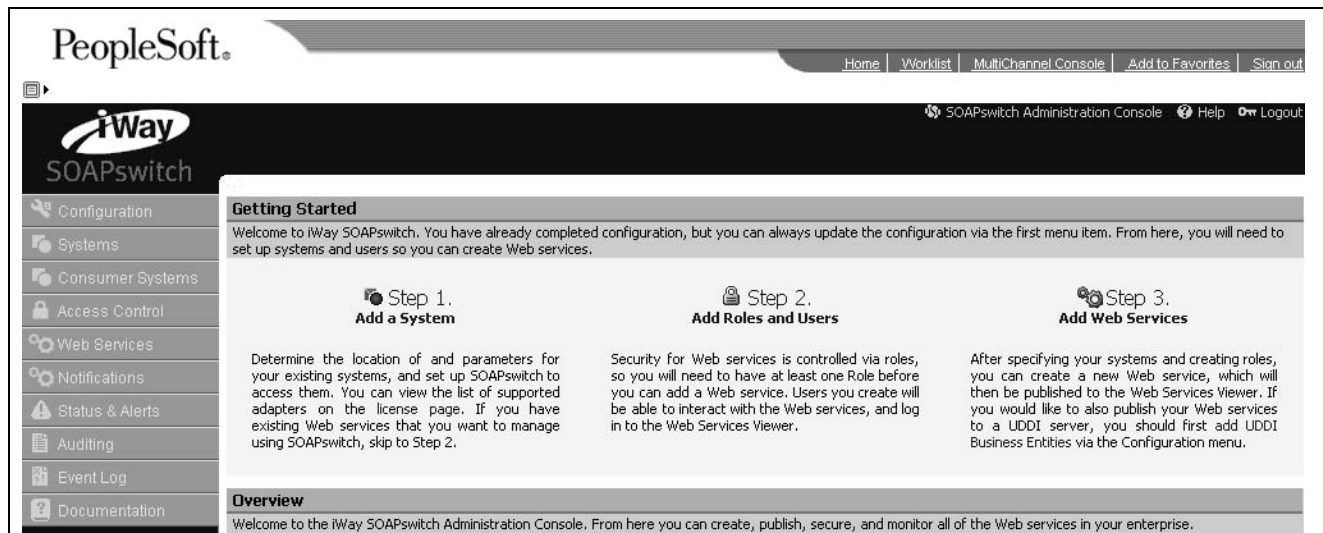
To change the iWay SOAPswitch login information:

1. To change the user ID:
 - a. Select the Change User ID box.
 - b. In the User ID field, enter a new user ID.
2. To change the password:
 - a. Select the Change Password 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.

Generating WSDL Using the ERP Connectors

You can generate WSDL from SAP, Oracle and Siebel systems using the delivered ERP connectors 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 16, “Creating Third-Party Integrations Using WSDL,” iWay SOAPswitch and Adapter Documentation, page 365.](#)

Generating WSDL for Outbound Transactions

The steps to generate WSDL using the ERP connectors delivered with PeopleTools for outbound transactions from PeopleSoft are:

1. Configure the appropriate ERP connector.
2. Add a system.
3. Add roles and users.
4. Add a web service to the system.

Generating WSDL for Inbound Transactions

The follow list shows the steps to generate WSDL using the ERP connectors delivered with PeopleTools for inbound transactions to PeopleSoft.

1. Configure the appropriate ERP connector.
2. Add a system.
3. Add a consumer system.
4. Add roles and users.
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.

You must configure an ERP connector for outbound and inbound transactions.

Please see the iWay SOAPswitch adapter documentation 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 16, “Creating Third-Party Integrations Using WSDL,” iWay SOAPswitch and Adapter Documentation, page 365.](#)

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.
 - Under the Getting Started section of the Administration Console, click Step 1. Add a System, or
 - 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”

Adding Roles and Users

Security for web services is controlled via roles. Users 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 you 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:
 - Under the Getting Started section of the Administration Console, click Step 2. Add Roles and Users, or
 - 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 displays. 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:
 - Under the Getting Started section of the Administration Console, click Step 3. Add Web Services, or
 - In the left navigation area, click Web Services.

After you publish the web service to PeopleSoft, in the Pure Internet Architecture the WSDL information page displays and is populated with the generated WSDL for you to import into PeopleSoft.

See [Chapter 16, “Creating Third-Party Integrations Using WSDL,” Importing WSDL Into PeopleSoft, page 372.](#)

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 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 you enter in the URL field must be the same as the integration gateway URL specified on the Gateways page—with the exception that the URL you enter in the destination definition is appended with the HTTP listening connector. For example, if the URL 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, click the Destinations tab. The Destination Definition Wizard displays.

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.

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.”

Importing WSDL Into PeopleSoft

You can import WSDL into PeopleSoft that you create with the ERP connectors, as well as WSDL that you create or access using different means. When you import WSDL into PeopleSoft, PeopleSoft Integration Broker adds it to the WSDL repository. This section discusses how to:

- Import WSDL that you generate using iWay SOAPswitch.
- Import WSDL from files and URLs.

Note. On DB2 and DB2/UNIX Unicode and non-Unicode databases, after you import WSDL into PeopleSoft you can view only up to 16 kilobytes of the raw WSDL in PeopleSoft, even though the actual amount of WSDL imported may be greater.

Importing WSDL Generated Using iWay SOAPswitch

When you use the iWay SOAPswitch Web Services Definition Wizard to create a web service or Notification Wizard to create a notification, iWay SOAPswitch automatically generates a WSDL document, opens the PeopleSoft Pure Internet Architecture and displays the WSDL Information page populated with the generated WSDL for you to import.

WSDL Information

WSDL Name:

WSDL Content:

CUSTOMER

```
<?xml version="1.0" encoding="UTF-8" ?
><wsdl:definitions name='CustomerServerImpl'
    targetNamespace='http://actional.com/exposed/CustServer/CustomerServerImpl'
    xmlns='http://actional.com/exposed/CustServer/CustomerServerImpl'
    xmlns:acb='http://actional.com/acb'
    xmlns:wsdl='http://schemas.xmlsoap.org/wsdl/'
    xmlns:soap='http://schemas.xmlsoap.org/soap/'
    xmlns:CustomerServerIm='http://actional.com/exposed/CustServer/CustomerServerImpl'
    xmlns:wsdl_target='http://actional.com/exposed/CustServer/CustomerServerImpl'
    xmlns:lang='http://actional.com/exposed/j
```

Import

WSDL Information page

To import the WSDL into PeopleSoft:

1. In the WSDL Name field, enter a proper name for the WSDL.
2. Click the Import button.

After you click the Import button, the WSDL Repository page displays, where you can inspect service details and create transaction metadata for the WSDL.

See [Chapter 16, “Creating Third-Party Integrations Using WSDL,” Creating Service Transaction Metadata for Imported WSDL, page 375.](#)

Importing WSDL from Files and URLs

To import WSDL from files and URLs, use the Import WSDL page in the Import WSDL component (WSDL_LOAD). To access this page, select PeopleTools, Integration Broker, Web Services, Import WSDL.

PeopleSoft Proprietary and Confidential

373

Import WSDL page

To import WSDL from a file or URL:

1. In the WSDL Name field, enter a unique name for the document.
2. Choose the import method.
 - a. To import WSDL from a URL, select URL, enter a URL and click the Load from URL button. The WSDL Content box displays the WSDL content.
 - b. To import WSDL from a file, click File, and click the Load from File button. A new page displays where you can browse and upload a file. Click the Browse button and navigate to the WSDL document to import. Click the Open button to load the file name into the Browse box. Click the Upload button. The Import File page appears with the WSDL contents displayed in the WSDL Content box.
3. Click the Import button.

You can now access the WSDL in the WSDL repository using the WSDL Repository page.

See [Chapter 16, “Creating Third-Party Integrations Using WSDL,” Using the WSDL Repository, page 374.](#)

Using the WSDL Repository

WSDL that you import into PeopleSoft is stored in the WSDL repository and accessible through the WSDL Repository page in the WSDL Repository component (WSDL_ANALYZER). To access the page, select PeopleTools, Integration Broker, Web Services, WSDL Repository.

Services | WSDL Document

WSDL Name: CUSTOMER

Service	Port	Endpoint	Service Detail
customerService	customer	http://MDAWSON051503:8080/ssw/runtime	Service Detail

Viewing service information for the CUSTOMER WSDL document in the WSDL Repository

When you select a WSDL document in the repository, use the Services tab to view high-level service information. Click the Service Detail link to view more detailed information about the service and to create transaction metadata using the WSDL Operation Wizard.

Use the WSDL Document tab to view a well-formed version of the WSDL.

See Also

[Chapter 16, “Creating Third-Party Integrations Using WSDL,” Creating Service Transaction Metadata for Imported WSDL, page 375](#)

Creating Service Transaction Metadata for Imported WSDL

PeopleSoft provides a WSDL Operation Wizard that guides you through the process of creating PeopleSoft Integration Broker transaction metadata from WSDL. You access the WSDL Operation Wizard from the WSDL repository.

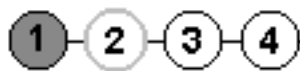
Use the WSDL Operation Wizard to:

1. Specify the node for the transaction.
2. Specify the request message and, depending on the transaction type, the response message.
3. Select a message channel.

The WSDL Operation Wizard allows you to create new node definitions, message definitions and message channel definitions directly from the wizard.

After you complete the wizard, PeopleSoft Integration Broker, creates the transaction. However, you must create any necessary transformations, create any required relationships and add the appropriate PeopleCode to message definitions to complete the integration.

Common Elements Used to Create Service Transaction Metadata



Shows the number of pages in the WSDL Operation Wizard.

The page with which you are currently working displays highlighted in orange. Pages you have completed display darkened.

< Previous

Click the Previous button to go back to the previous page in the wizard.

Next >

Click the Next button to proceed to the next page in the wizard.



Click the Finish button to generate the service transaction metadata.



Click the Cancel button to exit the page and cancel the current transaction metadata set up.

Accessing the WSDL Operation Wizard

To access the WSDL Operation Wizard:

1. Select PeopleTools, Integration Broker, Web Services, WSDL Repository.
2. Select the WSDL that contains the service for which you want to create PeopleSoft Integration Broker metadata.
 - a. Click the Search button to list all WSDL documents in the database.
 - b. Click the appropriate WSDL document. The WSDL Repository page displays.
3. In the Service Detail column, click the Service Detail link of the service with which to work. The Service Details page displays.
4. In the Routing Data column, click Create in the row that contains the operation for which you want to create metadata.

The WSDL Operation Wizard displays.

Specifying Nodes for Services

Use the first page in the WSDL Operation Wizard to specify a node for the service.

Step 1 of 4

1 2 3 4

WSDL Operation: getCreditCardData

Operation Type: Outbound Sync

Select a Node for the Service

☐ **Use Existing Node Definition**

Node Name:

☒ **Create a New Node Definition**

Node Information

*Node Name:

*Description:

*Authentication Option: ▼

*Password:

Confirm Password:

Page one of the WSDL Operation Wizard

The WSDL Operation Wizard enables you to select an existing node definition for the transaction, or create a new node definition.

If you select an existing node definition and the default target connector for the node is defined as something other than the HTTP target connector, the WSDL Operation Wizard will override the connector at the transaction level and use the HTTP target connector. If you create a new node definition, the default target connector is the HTTP target connector.

Using Existing Node Definitions

To use an existing node definition:

1. Select the Use Existing Node Definition option.
2. Click the Lookup button to select the node name.
3. Click the Next button to proceed to the next page in the WSDL Operation Wizard, where, depending on the transaction type, you select request and response messages for the service.

Creating New Node Definitions

To create a new node definition:

1. Select the Create a New Node Definition button.

2. In the Node Name field, enter a name for the new node definition.
3. In the Node Description field, enter a descriptive name for the node.
4. From the Authentication drop-down list, select an authentication method.

See [Chapter 9, “Configuring Nodes and Transactions,” Creating Node Definitions, page 196](#).

The valid options are:

- None. (Default)
 - Certificate
 - Password
5. (Optional) In the Password field, enter a password.
 6. (Optional) In the Confirm Password field, re-enter the password.
 7. Click the Next button to proceed to the next page in the WSDL Operation Wizard, where you select request and response messages for the service.

Selecting Request and Response Messages for Services

Use the second page in the WSDL Operation Wizard to select the request and response messages to use for the node transactions.

If the node transaction is synchronous, you must specify a request and response message. If the transaction is asynchronous, you need only select a request message.

You can select rowset-based or nonrowset-based messages, as long as you have created the appropriate transformations to transform the incoming SOAP messages into the proper format, and to transform the outgoing messages into SOAP format.

Step 2 of 4

1 2 3 4

< Previous Next > Finish Cancel

WSDL Operation: getCreditCardData

Operation Type: Outbound Sync

Select Message Names for the Service

Request Message

☐ Use Existing Message

Message Name:

Message Version:

☒ Create a New Message

Message Name:

Message Version:

Response Message

☒ Use Existing Message

Message Name:

Message Version:

☐ Create a New Message

Message Name:

Message Version:

< Previous Next > Finish Cancel

Page two of the WSDL Operation Wizard

Request and response messages must reside in the same message channel.

If you are creating new messages for both the request and response, you will be prompted to create the message channel for the messages on the next page in the wizard or you can use the delivered WEB_SERVICE channel. If you are using one existing message and creating one new message, PeopleSoft Integration Broker automatically assigns the new message to the same message channel as the existing message you chose.

See [Chapter 16, “Creating Third-Party Integrations Using WSDL,” Selecting Message Channels for Service Messages, page 380.](#)

When you have specified the request and response messages for the service, click the Next button to proceed to the next page in the wizard.

Using Existing Messages

To use an existing request or response message:

1. Select the Use Existing Message option in the appropriate section.

To specify an existing request message, select the option in the Request Message section. To specify an existing response message, select the option in the Response Message section.

2. In the Message Name field, click the Lookup button to select a message that has been defined for the node.
3. In the Message Version field, click the Lookup button to select a message version.

Creating New Messages

To create new request or response message:

1. Select the Create a New Message option in the appropriate section.

To create a new request message, select the option in the Request Message section. To create a new response message, select the option in the Response Message section.

2. In the Message Name field, enter the name for the new message.

Newly created messages default to *VERSION_1*. PeopleSoft Integration Broker populates the Message Version field with this value.

Selecting Message Channels for Service Messages

After you specify request and response messages for a service, you must select the message channel for them.

Step 3 of 4

1

2

3

4

< Previous

Next >

Finish

Cancel

WSDL Operation: getCreditCardData

Operation Type: Outbound Sync

Select a Message Channel for the Service

☒

Use existing Message Channel

Channel Name:

☐

Create a new Message Channel

Channel Name:

< Previous

Next >

Finish

Cancel

Page three of the WSDL Operation Wizard

You can edit this page only if:

- You are working with an asynchronous transaction and you created a new message for the service on the previous page of the wizard, or
- You are working with a synchronous transaction and you created a new request and a new response message for the service on the previous page of the wizard.

In these cases you can choose to assign the message or messages to an existing message channel, or create a new message channel.

In all other situations PeopleSoft Integration Broker automatically populates the message channel name on this page, and the page is read-only with all options disabled. The following bullet points explain how PeopleSoft Integration Broker derives the channel name in these situations.

- When you are working with an asynchronous transaction, and you chose an existing message for the service on the previous page of the wizard, by default PeopleSoft Integration Broker uses the existing message channel for that message.
- When you are working with a synchronous transaction and:
 - You chose an existing request message *and* an existing response message for the service on the previous page of the wizard. In this case the name of the message channel to which both messages belong displays.
 - You chose one existing message and created one new message on the previous page of the wizard. In this case PeopleSoft Integration Broker automatically assigns the new message to the same message channel as the existing message you chose, and that channel name displays.

Using Existing Message Channels for Service Messages

To use an existing message channel:

1. Select the Use Existing Message Channel option.
2. Click the Lookup button to select a message channel from the database.
3. Click the Next button to proceed to the next page in the WSDL Operation Wizard.

Creating New Message Channels for Service Messages

To create a new message channel:

1. Select the Create a New Message Channel option.
2. Enter a name for the message channel
3. Click the Next button to proceed to the next page in the WSDL Operation Wizard.

Generating Service Transaction Metadata

The last page in the WSDL Operation Wizard enables you to review the values you have specified in the wizard and then generate the service transaction metadata.

Step 4 of 4

1 2 3 4

< Previous Next > Finish Cancel

WSDL Operation: getCreditCardData

Operation Type: Outbound Sync

Node Name: QE_WSDL_DEMO

Channel Name: QE_SALES_ORDER_SYNC_CHNL

Request Message Name: QE_OUTBOUND_SYNC_MSG

Request Message Version: VERSION_1

Response Message Name: QE_RESPONSE_MSG

Response Message Version: VERSION_1

< Previous Next > Finish Cancel

Page four of the WSDL Operation Wizard

To generate the service transaction metadata, click the Finish button.

A message displays indicating that the operation was added to the node, and the Service Details page displays.

Note. When you attempt to exit the WSDL Operation Wizard, a message displays that says there is unsaved data on the page. Disregard this message and click the Continue button to exit.

Viewing Generated Service Transaction Metadata

After you use the WSDL Operation Wizard, you can navigate to the appropriate PeopleSoft Application Designer and the PeopleSoft Pure Internet Architecture page to view any new node, message and message channel object definitions you created.

When you view the node definition, you will notice the following values that PeopleSoft Integration Broker has populated. To access the node definition, select PeopleTools, Integration Broker, Node Definitions.

- If you created a new node, it displays in the search list.
- Select the new definition and click the Node tab. The description and authentication option you created displays.
- Click the Transaction tab. A transaction exists for the outbound message. Click the Edit link to view the following:
 - The Transaction Detail tab shows that the transaction is active. In addition, the Connector Override option is enabled and the HTTP target connector selected.
 - The Message tab shows the inbound and outbound messages.

- The Connectors tab shows the SOAPAction property set to the same value shown on the Service Details page for the operation. In addition, the Primary URL property value is the same value as that shown for the endpoint on the Service Details page.

If you created new messages or new message channel, you can access the new definitions in PeopleSoft Application Designer.

If you view these definitions, you will note that they are active.

Completing Integrations

After you have completed using the WSDL Operation Wizard, you must perform the following steps to complete the integration:

1. Create a transformation for the imported message.
2. Create a relationship to consume the service.
3. Add the appropriate subscription OnRequest PeopleCode to the message for inbound integrations or create the client invocation PeopleCode for outbound integrations.

See Also

[Chapter 13, “Administering Relationships,” Configuring Relationships, page 277](#)

[Chapter 12, “Applying Filtering, Transformation and Translation,” Applying Transformations, page 257](#)

Generating WSDL from Component Interfaces and Messages

PeopleSoft Integration Broker enables you to generate WSDL documents from PeopleSoft component interface and message object definitions. To generate WSDL for these PeopleSoft object definitions, use the Published EIPs page in the Published EIPs component (WSDL_APPMSG). To access this page, select PeopleTools, Integration Broker, Web Services, Published EIPs.

Published EIPs

☒ **Component Interface**

Component Interface

Web Service: 

Description:

☐ **Message**

Message

Node Name:

Message Name:

Published EIPs page

Generating WSDL from Component Interfaces

Before you generate WSDL from a component interface, verify that the SOAPTOCI message is active and that you have added security to the SOAPTOCI Web Library.

To generate WSDL from a component interface:

1. Select PeopleTools, Integration Broker, Web Services, Published EIPs.
2. Select the Component Interface option.
3. In the Web Service field, use the Lookup button and select a component interface.
The component interface object name and description, if one exists, populates the Web Service and Description fields.
4. Click the Search button. A Web Services to CI section displays at the bottom of the page and lists items that match your search.
5. Click the WSDL link in the desired row.
6. A new web browser window opens and displays the component interface as a WSDL document.

Generating WSDL from PeopleSoft Messages

To expose a PeopleSoft message as an integration point, it must first be associated with a node transaction. Node transactions contain all the relevant data necessary to generate a proper WSDL document—transaction types, direction (inbound versus outbound), and so forth. As a result, when WSDL is generated for a message, it is ultimately being generated from its corresponding node transaction. Without it, the WSDL would be incomplete.

Understanding Endpoints for Published PeopleSoft Node Transactions

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.
```

Understanding Generating WSDL for Nonrowset-based Messages

If you choose a nonrowset-based message to export there is no valid structure to which to refer to build an XML schema. As a result, all of the WSDL bindings, routings and so forth are generated, but no XML tags or field name definitions are generated for the node transaction. You must insert that information into the WSDL.

The following example shows generated WSDL for a nonrowset-based PeopleSoft message, QE_FLIGHTPLAN_UNSTRUCT_SYNC.

```
<?xml version="1.0" ?>
<wSDL:definitions name="QE_FLIGHTPLAN_UNSTRUCT_SYNC" xmlns:soap=
  "http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:wSDL="http://schemas.xmlsoap.org/wsdl/" targetNamespace=
    "http://peoplesoft.com/QE_FLIGHTPLAN_UNSTRUCT_SYNCSoapIn"
  xmlns:wSDL_target="http://peoplesoft.com/QE_FLIGHTPLAN_UNSTRUCT_SYNCSoapIn"
  xmlns:QE_FLIGHTPLAN_UNSTRUCT_SYNCRequest="http://peoplesoft.com
    /QE_FLIGHTPLAN_UNSTRUCT_SYNCRequest"
  xmlns:QE_FLIGHTPLAN_UNSTRUCT_SYNCResponse="http://peoplesoft.com
    /QE_FLIGHTPLAN_UNSTRUCT_SYNCResponse">
```

```

<wsdl:documentation />
<wsdl:types>
  <xsd:schema xmlns="http://peoplesoft.com/QE_FLIGHTPLAN_UNSTRUCT_
    SYNCRequest" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://peoplesoft.com/QE_FLIGHTPLAN_
    UNSTRUCT_SYNCRequest">
    <xsd:element name="QE_FLIGHTPLAN_UNSTRUCT_SYNC" type=
      "QE_FLIGHTPLAN_UNSTRUCT_SYNCType" />
    <xsd:complexType name="QE_FLIGHTPLAN_UNSTRUCT_SYNCType">
      <!--
        Could not create schema for PeopleSoft nonrowset-based message.

      -->
    </xsd:complexType>
  </xsd:schema>
  <xsd:schema xmlns="http://peoplesoft.com/QE_FLIGHTPLAN_UNSTRUCT_SYNCResponse
    " xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://peoplesoft.com/QE_FLIGHTPLAN_UNSTRUCT_SYNCResponse">
    <xsd:element name="QE_FLIGHTPLAN_UNSTRUCT_SYNC" type="QE_FLIGHTPLAN_UNSTRUCT_
      SYNCType" />
    <xsd:complexType name="QE_FLIGHTPLAN_UNSTRUCT_SYNCType">
      <!--
        Could not create schema for PeopleSoft nonrowset-based message.

      -->
    </xsd:complexType>
  </xsd:schema>
</wsdl:types>
<wsdl:message name="QE_FLIGHTPLAN_UNSTRUCT_SYNCSoapIn">
  <wsdl:part name="parameters" element="QE_FLIGHTPLAN_UNSTRUCT_SYNCRequest
    :QE_FLIGHTPLAN_UNSTRUCT_SYNC" />
</wsdl:message>
<wsdl:message name="QE_FLIGHTPLAN_UNSTRUCT_SYNCSoapOut">
  <wsdl:part name="parameters" element="QE_FLIGHTPLAN_UNSTRUCT_SYNCResponse:
    QE_FLIGHTPLAN_UNSTRUCT_SYNC" />
</wsdl:message>
<wsdl:portType name="QE_FLIGHTPLAN_UNSTRUCT_SYNC">
  <wsdl:operation name="QE_FLIGHTPLAN_UNSTRUCT_SYNCSoapIn">
    <wsdl:output message="wsdl_target:QE_FLIGHTPLAN_UNSTRUCT_SYNCSoapOut" />
    <wsdl:input message="wsdl_target:QE_FLIGHTPLAN_UNSTRUCT_SYNCSoapIn" />
  </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="QE_FLIGHTPLAN_UNSTRUCT_SYNCBinding" type="wsdl_target
  :QE_FLIGHTPLAN_UNSTRUCT_SYNC">
  <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap⇒
  /http" />
  <wsdl:operation name="QE_FLIGHTPLAN_UNSTRUCT_SYNCSoapIn">
    <soap:operation soapAction="QE_FLIGHTPLAN_UNSTRUCT_SYNCSoapIn" />
    <wsdl:output>

```

```

        <soap:body use="literal" />
    </wsdl:output>
    <wsdl:input>
        <soap:body use="literal" />
    </wsdl:input>
</wsdl:operation>
</wsdl:binding>
<wsdl:service name="QE_FLIGHTPLAN_UNSTRUCT_SYNCService">

    <wsdl:documentation>PeopleSoft, localhost.PT_LOCAL, OS, QE_FLIGHTPLAN_
    UNSTRUCT_SYNC.VERSION_1, QE_FLIGHTPLAN_UNSTRUCT_SYNC.VERSION_1</wsdl:
    documentation>
    <wsdl:port name="QE_FLIGHTPLAN_UNSTRUCT_SYNC" binding="wsdl_target:
    QE_FLIGHTPLAN_UNSTRUCT_SYNCBinding">
        <soap:address location="" />
    </wsdl:port>
</wsdl:service>
</wsdl:definitions>

```

Generating WSDL from Messages

To generate WSDL from a message:

1. Access the Published EIPs page.
2. Select the Message option.
3. In the Node Name field, click the Lookup button to select a node name.
4. In the Message Name field, click the Lookup button to select a message name.
5. Click the Search button.

An EIPs as WSDL section displays at the bottom of the page and lists items that match your search.

6. Click the WSDL link in the desired row.
7. A new web browser window opens and displays the node transaction as a WSDL document.

CHAPTER 17

Setting Up Secure Messaging Environments

This chapter discusses how to:

- Work with digital certificates.
- Implement authentication and nonrepudiation.

Understanding Secure Messaging

This provides an overview of types of messaging security, digital certificates and nonrepudiation.

Types of Messaging Security

Secure messaging with PeopleSoft Integration Broker can include three types of security:

Authentication.	This is the ability to securely identify the sender of a message. It can be accomplished using a password or a digital signature.
Nonrepudiation.	This is an enhanced form of digital signature authentication with two benefits. It ensures that neither participant in a transaction can disavow participation, and it guarantees the integrity of the message content.
Encryption.	This type of security supports privacy. The sender translates the content of a message into a secret code that only the receiver can decrypt. PeopleSoft Integration Broker employs the SSL protocol for encryption.

See Also

Chapter 17, “Setting Up Secure Messaging Environments,” Implementing Authentication and Nonrepudiation, page 394

Digital Certificates

All of these security types (except password authentication) use 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 message to verify the sender’s identity and to provide the recipient with the means to encode a response.

Digital signature authentication and nonrepudiation both use digital certificates to provide the keys to generate the necessary signatures. You use PeopleSoft Security to install digital certificates in each node’s application server keystore to support these features.

SSL implements encryption using a public key infrastructure (PKI), for which the digital certificates provide the keys. When you use an HTTP listening or target connector to exchange messages with a third-party system or a remote gateway across a firewall, SSL encryption is not implemented by default. To implement SSL for these cases, you must install digital certificates on the participating gateways.

Note. SSL on an integration gateway incorporates an authentication element, which is different from message authentication at a PeopleSoft Integration Broker node. SSL authentication does not need to be configured separately. Consequently, in this PeopleBook, the term *SSL encryption* refers to the combined elements of SSL, including SSL authentication.

Implementing SSL encryption for messaging requires installing two sets of digital certificates to support the two complementary parts of SSL:

- SSL with server authentication, which secures inbound messages.

You use software supplied with your BEA WebLogic or IBM WebSphere web server to install digital certificates in the web server's keystore.

- SSL with client authentication, which secures outbound messages.

You use a Java command line utility provided with the gateway's PeopleSoft Pure Internet Architecture installation to install digital certificates in the gateway's keystore.

See Also

Chapter 17, "Setting Up Secure Messaging Environments," Working With Digital Certificates, page 389

Nonrepudiation

PeopleSoft Integration Broker applies nonrepudiation to cross-node messaging by digitally signing request messages and their responses.

In PeopleSoft applications, nonrepudiation provides two-way protection; both the request message and its response are nonrepudiated. PeopleSoft Integration Broker uses PKI technology to implement nonrepudiation for messaging. Each participating node's keystore contains its own private key and the public keys of the nodes with which it exchanges nonrepudiation messages.

Here's how nonrepudiation works:

1. Node A generates a number, known as a *digest*, which uniquely identifies its request message.
2. Node A uses its private key to generate a signature based on the digest, and inserts the signature into the nonrepudiation request message.
3. Node A sends the nonrepudiation request message to node B.
4. When the nonrepudiation request message is received, 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 message, and compares it to the received digest to confirm the message integrity.

5. Node B generates a digest that uniquely identifies its response message.
6. Node B uses its private key to generate a signature based on the digest, and it inserts the signature into the nonrepudiation response message to confirm receipt of the nonrepudiation request message.
7. Node B sends the nonrepudiation response message to node A.

8. When the nonrepudiation response message 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 message and compares it to the received digest to confirm the message integrity.

Nonrepudiation produces the following results:

- The sending node cannot repudiate that the message was sent, because the receiving node has a copy of the request signed by the sender.
- The receiving node cannot repudiate that the message was received and processed, because the sending node has a copy of the response signed by the receiver.
- The message integrity is verified, because the validated signature of each nonrepudiation message assures that the message content as received exactly matches the content as sent.

Working With Digital Certificates

This section provides overviews of certificate authorities and certificate installation elements and discusses how to:

- Access certificate properties.
- Export and convert certificates.

Understanding Certificate Authorities

A certificate authority (CA) is a trusted third-party organization or company that issues digital certificates used to create digital signatures and encryption keys. The role of the CA in this process is to guarantee the identity of the party granted the certificate. Usually, this means that the CA has an arrangement with a financial institution that provides information to validate the grantee's identity.

To install digital certificates for secure messaging, you must select a CA from whom to obtain the certificates. There are many CAs to choose from, and most of them do business on the World Wide Web. Some of the best known are:

- Verisign, Inc.
- Entrust Technologies.
- Baltimore Technologies.
- Thawte.

There are also numerous lesser known CAs, which might be appropriate if they are well known in a particular geographical region or industry. One of the systems participating in a secure integration might even serve as CA for the other participants. Each CA provides a unique set of security services and has its own way of handling digital certificates.

Before you implement secure messaging with PeopleSoft Integration Broker, investigate the available CAs, select one or more from whom you will obtain digital certificates, and familiarize yourself with their policies and procedures.

Understanding 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, <i>QE_LOCAL</i> .
Organization unit (OU)	The part of the organization to which the entity belongs; for example, <i>Accounts Receivable</i> .
Organization (O)	The name of the organization or company; for example, <i>PeopleSoft</i> .
Locality (L)	The city or equivalent locality of the organization; for example, <i>Pleasanton</i> .
State (ST)	The state, province, or equivalent region of the locality; for example, <i>California</i> .
Country (C)	The country of the locality; for example, <i>US</i> .

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
jQAwgYKCGyEApGAHNBjuByh8qXFCz33TgUjRm8S6ti7fw23rKWyipQ0VgqeAD6eHr0pini
```



```

lyJPP0iJJ5fY0h2h78h0r8o+nJosTcqZL3jP+rSVick7qPPyXjcxP1UCGz/8RnykFDnbwjziwi+p
Mesowa8hfBss0ga2ZzsmLV8Q4SyYE3UCAwEAAaAAMA0GCSqGSIb3DQEBAUAA4GBACT0owTCngrU
/HAMAZgT/206hiZaD40VBrgLYzmRvUiVhK0yTUzUv57ks7U6DQYt+rnWwNjTVbeAq05eZiT7hXbj
Pwl8lGj+Adb6FGY0t40hicZ0gNMHtURVop6iNJ9scx0mVcPk00yX5f1rWfDZ0KZrWZSFGI6Lwdud
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-----
MIICIDCCACqgAwIBAgIQrDVQJKAACKLR0/bIDJMSVDANBgkqhkiG9w0BAQQFADBy
MQswCQYDVQQGEwJVUZELMAKGA1UECBMCQ0ExEzARBgNVBAcTClBsZWZzYW50b24x
FzAVBgNVBAoTDlBlb3BsZVNvZnQgSW5jMRMwEQYDVQQLEwpQZW9wbGVub29sMRMw
EQYDVQQDEwpQZW9wbGVub29sMB4XDTAwMDMxMDIxMTIzNVoxDTA1MDMxMDIxMTIz
NVowcjELMAKGA1UEBhMCVVMxMzA1BgNVBAGTAkNBMRMwEQYDVQQHEwpQbGVhc2Fu
dG9uMRcwFQYDVQQKEw5QZW9wbGVub29sIEluYzETMBEGA1UECXMKUGVvcGx1VG9v
bDEtMBEGA1UEAxMKUGVvcGx1VG9vbDBcMA0GCSqGSIb3DQEBAQUAA0sAMEgCQQCy
o44wplb57M272GRP3sC4TtLm/MD1G9osRjG9BWnsjjTij9GNI6Rnf9cNxxkj+AGQY
gnE3P7lp9rYN6GQxPlDnAgMBAAGjPDA6MASGA1UdDwQEAwIBxDAMBGNVHRMEBTAD
AQH/MB0GA1UdDgQWBBSkFZJ1Dtt5uE6muLRN3rwrPsUCsTANBgkqhkiG9w0BAQQF
AANBAJec3hFPS2SLSDtflI9mSA7UL1Vgbxr5zZ4Sj9y4I2rncrTwcBqj7EBp9n/Z
U/EwDEljVbE8SSDYr1Emgoxsr4Y=
-----END CERTIFICATE-----

```

Root Certificate

The root certificate contains the CA's digitally signed public key. It's also known as a *chain file* or a *signer certificate*. The process of obtaining a root certificate from a CA depends on the CA. The CA typically sends an email with the certificate or requires you to download it from a specified page.

The signed public key certificate also contains an embedded copy of the CA's root certificate, which you can export.

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 NT or Windows 2000 machine to access the certificate properties dialog box.

To access certificate properties:

1. Double-click any certificate file with a .DER (binary format) extension or a .CER (PEM format) extension.

This invokes the Windows extensions for security management, which open a dialog box so you can inspect the certificate properties.

2. (Optional) Access the properties of the embedded root certificate.

a. Select Certification Path.

A tree structure appears, showing the hierarchical chain of trust between the public key certificate and its issuer root certificate. Your certificate has the common name that you supplied for it, and the issuer root certificate (its parent) has the name of its issuing CA.

b. Select the root certificate, and click View Certificate.

A dialog box displays 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 17, "Setting Up Secure Messaging Environments," Accessing Certificate Properties, page 391](#).

2. In the certificate properties, select Details, then click Copy to File.

The Certificate Export Wizard launches.

3. Click Next, then select a format.

Base64-encoded X.509 (.CER) is the PEM format option, which is recommended. The *DER encoded binary X.509 (.CER)* option may also work, depending on the environment.

4. Click Next, and then browse to select a location and file name for the new certificate file.

Specify the same location as the certificate. Ideally, you should give an exported root certificate file the same name as the issuing CA.

5. Click Next, then Finish to save the root certificate file.

A message indicates when the export is successful.

To export a certificate from an application server keystore:

1. In the PeopleSoft Pure Internet Architecture, sign on to the application database and select PeopleTools, Security, Security Objects, Digital Certificates.

The Digital Certificates page appears.

2. Click the Detail link of the desired certificate, then click Export.

The Export Certificate page appears, containing the exportable certificate content in a long edit box.

3. Copy the entire certificate content and sign out of the database.

Note. Save this certificate content to a file with a .CER extension.

The following example shows the Certificate Detail page:

Certificate Detail - QE_LOCAL

Subject Information

Common Name: QE_LOCAL

Org Unit: PeopleTools

Organization: PeopleSoft

Locality: Pleasanton

State/Province: CA **Country:** US

Issuer Information

Common Name: PeopleTool

Org Unit: PeopleTool

Organization: PeopleSoft Inc

Locality: Pleasanton

State/Province: CA **Country:** US

Certificate Information

Serial Number: 0F:27:B9:18:00:00:06:68

Fingerprint: 9E:1F:40:D3:2C:9B:75:9B:E7:2E:A4:68:98:30:C5:B1

Valid from 08/22/2002 14:11:12 **to** 08/22/2003 14:11:12

Algorithm: MD5 with RSA encryption

Description:

Version: 3
Serial number: 1092044947390989928
Signature algorithm: md5WithRSAEncryption
Issuer: cn=PeopleTool,ou=PeopleTool,o=PeopleSoft
Inc,l=Pleasanton,st=CA,c=US
Valid not before: Thu Aug 22 14:11:12 PDT 2002

Return

Renew

Export

Certificate Detail page for QE_LOCAL node

Implementing Authentication and Nonrepudiation

This section discusses how to:

- Install node-based digital certificates.
- Configure authentication.
- Configure nonrepudiation.

Note. PeopleSoft Integration Broker uses node-based digital certificates to generate digital signatures for authentication and nonrepudiation, but not encryption. External SSL encryption is provided by the digital certificates that you install separately on the gateway's web server.

Installing Node-Based Digital Certificates

At each node that participates in authenticated or nonrepudiated transactions, you must install digital certificates in the node's application server keystore. Any participating third-party system must have a set of certificates complementary to those installed at the PeopleSoft nodes. You use the PeopleSoft Security Digital Certificates page to install the certificates, as shown here:

Digital Certificates				Customize Find 		First	1-16 of 16	Last
Type	*Alias	*Issuer Alias	Valid to					
Root CA	GTE CyberTrust Global Root	GTE CyberTrust Global Root		Detail	+	-		
Root CA	GTE CyberTrust Root	GTE CyberTrust Root		Detail	+	-		
Root CA	KeyWitness Root	KeyWitness Root		Detail	+	-		
Root CA	Root SGC Authority	Root SGC Authority		Detail	+	-		
Root CA	Thawte Personal Basic	Thawte Personal Basic		Detail	+	-		
Root CA	Thawte Personal Premium	Thawte Personal Premium		Detail	+	-		
Root CA	Thawte Premium Server	Thawte Premium Server		Detail	+	-		
Root CA	Thawte Server	Thawte Server		Detail	+	-		
Root CA	Verisign Class 1	Verisign Class 1		Detail	+	-		
Root CA	Verisign Class 1 - G2	Verisign Class 1 - G2		Detail	+	-		
Root CA	Verisign Class 2	Verisign Class 2		Detail	+	-		
Root CA	Verisign Class 2 - G2	Verisign Class 2 - G2		Detail	+	-		
Root CA	Verisign Class 3	Verisign Class 3		Detail	+	-		
Root CA	Verisign Class 3 - G3	Verisign Class 3 - G3		Detail	+	-		
Root CA	Verisign Class 4	Verisign Class 4		Detail	+	-		
Root CA	Verisign/RSA Secure Server CA	Verisign/RSA Secure Server CA		Detail	+	-		

Digital Certificates page

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 authenticated or nonrepudiated messaging.

Each of these certificates is stored in a record of type *Remote*.

Note. Each remote certificate is a copy of the local node certificate this is installed on the remote node that it represents.

Consequently, 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*.

Installing Root Certificates

You can use one of the existing root certificates delivered with the PeopleSoft application, or you can install a new one from a different CA. Either way, you must have a CA's root certificate installed before you can obtain the default local node's signed public key certificate from that CA.

To install a new root CA certificate:

1. Obtain a root certificate from the selected CA.

You can obtain the certificate in several ways:

- Go to the CA's web site and request a signed, PEM-formatted copy of the CA's root certificate.

The process of obtaining a root certificate from a CA depends on the CA.

- Export the root certificate from a public key certificate previously issued by that CA, or export it from an existing root CA row in another PeopleSoft application server keystore.

See [Chapter 17, "Setting Up Secure Messaging Environments," Exporting and Converting Certificates, page 392](#).

2. In the PeopleSoft Pure Internet Architecture, select PeopleTools, Security, Security Objects, Digital Certificates.
3. On the Digital Certificates page, add a new row, and select the type *Root CA*.
4. In the Alias field, enter the name of the CA, then click the lookup button for the Issuer Alias field. Both fields should have the same value, and the new certificate's row should contain an Add Root link.
5. Click Add Root.

The Add Root Certificate page appears.

6. Open the downloaded or exported root certificate file in a text editor and copy its certificate content, including the -----BEGIN CERTIFICATE----- and -----END CERTIFICATE----- lines.
7. Paste the copied certificate content into the long edit box on the Add Root Certificate page and click OK.

The Digital Certificates page reappears. The new certificate's row now contains a Detail link, which you can use to examine the certificate.

Installing Signed Public Key Certificates

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 install a signed public key certificate:

1. On the Digital Certificates page, add a new row, and select the type *Local Node*.
2. In the Alias field, enter the name of the PeopleSoft Integration Broker default local node.

You do not need to configure the default local node yet; however, you must use this name when you do configure it.

See [Chapter 9, “Configuring Nodes and Transactions,” Configuring Nodes, page 196](#).

3. Select the issuer alias of the root CA certificate whose CA you want to sign the local node certificate. The new row now contains a Request link.

Click the Request link to open the Request New Certificate page. You use the Request New Certificate page, shown here, to import the new local node certificate:

Request New Certificate

Subject Information

Common Name:

Org Unit:

Organization:

Locality:

State/Province:
Country:

Key Pair Information

Algorithm:

Key Size:

Additional Certificate Attributes

Email Address:

Challenge Pswd:

Request New Certificate page

To obtain and import the new local node certificate:

1. On the Digital Certificates page, click the Request link for the new certificate.

The Request New Certificate page appears.

2. Enter the subject information in the fields provided.

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 them. The DN will also be 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.

See [Chapter 17, “Setting Up Secure Messaging Environments,” Understanding Certificate Installation Elements, page 390](#).

3. Select the key encryption algorithm that this certificate uses, or accept the default value.
4. Select the key size that this certificate uses:
 - A 1024-bit key provides 128-bit encryption.
 - A 512-bit key provides 40-bit encryption.
5. (Optional) Enter an email address.
This information is also stored on the Detail page of the certificate as a DN attribute called *mail*.
6. Click OK to generate a new CSR.

The Certificate Signing Request page appears, containing the CSR in a long edit box.

Note. 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.

7. Copy the certificate request portion of the text.
Include the `-----BEGIN NEW CERTIFICATE REQUEST-----` and `-----END NEW CERTIFICATE REQUEST-----` lines.

Note. Save this CSR content to a text file.

8. Click OK.
The Digital Certificates page reappears. The new row now contains an Import link.
9. To obtain your local node certificate, submit the CSR to the CA that issued the selected root certificate.
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.
10. Open the saved certificate file in a text editor, then highlight and copy its entire contents, including the `-----BEGIN CERTIFICATE-----` and `-----END CERTIFICATE-----` lines.
11. Return to the Digital Certificates page and click the Import link for the new certificate.
The Import Certificate page appears.
12. Paste the copied certificate content into the long edit box and click OK.

Three outcomes are possible.

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 the Valid From field in the certificate's properties dialog box to determine when the certificate can be imported.

See [Chapter 17, "Setting Up Secure Messaging Environments," Accessing Certificate Properties, page 391](#).

Alternatively, 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 node certificate in your keystore must be accompanied by a root CA certificate signed by the same CA. See the following section to resolve this root certificate mismatch.

The third possibility is that the Digital Certificates page reappears 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.

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 17, "Setting Up Secure Messaging Environments," Exporting and Converting Certificates, page 392.](#)

2. Define a new root CA certificate in the keystore.

Refer to the previous procedure for establishing a root certificate.

3. Delete the local node row from the keystore's Digital Certificates page.

4. Add a new local node certificate to the keystore using the same issuer alias as the new root CA certificate.

Refer to the previous steps for installing a signed public key certificate.

Installing Remote Certificates

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.

The following requirements apply:

- The remote system's local node certificate must already be installed.

Refer to the previous steps for installing a signed public key certificate.

- The local system must have a root certificate installed with the same issuer alias (and actual issuer) as the remote system's local node certificate.

Refer to the previous steps for establishing a root certificate.

Note. For the purposes of this discussion, assume that both local and remote nodes are PeopleSoft applications. If the remote node is a third-party system, the same requirements must still be satisfied—the third-party system must provide a copy of its signed public key certificate to the PeopleSoft node.

To install a remote certificate:

1. Export a copy of the remote system's installed local node certificate.

See [Chapter 17, "Setting Up Secure Messaging Environments," Exporting and Converting Certificates, page 392](#).

2. In the PeopleSoft Pure Internet Architecture, sign on to the local database and select PeopleTools, Security, Security Objects, Digital Certificates.

The Digital Certificates page appears.

3. Add a new row, and select the type *Remote*.

4. In the Alias field, enter the name of the remote node.

You do not need to define the remote node yet; however, when you do define it, you must use the name that the remote system uses for its default local node.

See [Chapter 9, "Configuring Nodes and Transactions," Configuring Nodes, page 196](#).

5. Select the issuer alias of the remote system's local node certificate.

The new certificate's row now contains an Import link.

6. Click Import.

The Import Certificate page appears.

7. Paste the content of the exported public key certificate into the long edit box and click OK.

The Digital Certificates page reappears. The new certificate's row now contains a Detail link.

See Also

Enterprise PeopleTools 8.45 PeopleBook: Security Administration, "Setting up Digital Certificates and Single Signon," Working With Digital Certificates

Configuring Authentication

You can implement authentication functionality with passwords or digital signatures.

Password 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 9, "Configuring Nodes and Transactions," Creating Node Definitions, page 196](#).

Digital Signature Authentication

Digital signature authentication involves the following tasks:

- You must supply the digital certificates containing the private and public keys required for authenticated transactions.

These elements are required at every node that participates in an authenticated transaction; PeopleSoft Integration Broker handles the mechanics of applying the keys.

- 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 9, “Configuring Nodes and Transactions,” Creating Node Definitions, page 196](#).

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 in the properties of a message definition to indicate that it’s a nonrepudiation message.

When you do this on one system, you must select this option on the same message definition on every system that will participate in transactions using this message.

- 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 messages for future reference, you must make sure that they are archived.

With both archived and active nonrepudiation messages, you can regenerate the digest in PeopleSoft Integration Broker 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 messages it sends, and properly handling signatures in messages it receives.

See Also

[Chapter 17, “Setting Up Secure Messaging Environments,” Installing Node-Based Digital Certificates, page 394](#)

[Chapter 6, “Using Listening Connectors and Target Connectors,” Complying With Message Formatting and Transmission Requirements, page 92](#)

[Chapter 7, “Defining Message Channels and Messages,” page 137](#)

[Chapter 9, “Configuring Nodes and Transactions,” Configuring Nodes, page 196](#)

[Chapter 14, “Using Integration Broker Monitor,” page 291](#)

Setting Up SSL Encryption

This section discusses:

- Set up digital certificates for server authentication SSL for BEA WebLogic.
- Set up digital certificates for server authentication SSL for IBM WebSphere.
- Set up digital certificates for server authentication on application servers.
- Install digital certificates for client authentication SSL.
- Configure SSL encryption.

Understanding Installing Digital Certificates for Server Authentication SSL

You use utilities provided with the BEA WebLogic or IBM WebSphere server software to install web server-based certificates for SSL with server authentication. This authentication secures inbound messages. The web server requires three elements:

- The web server's private key.
- A certificate containing the web server's public key, digitally signed by a trusted certificate authority (CA).
- A root certificate from the CA that signed the web server's public key.

The information in section outlines the basic steps required to obtain and install the certificates and keys that you need. Because both WebLogic and WebSphere are third-party proprietary products—and each provides its own interface and methodology for establishing server authentication SSL—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 Server Authentication SSL on BEA WebLogic

This section describes how to install digital certificates for server authentication SSL 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.
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 Microsystems'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.
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----- MIIBtDCCAR0CAQAwDELMakGA1UEBhMCVVMxEDA0Bg⇒
NV
BAgtB0FyaXpnbmExEDA0BgnVBACTB1Bob2VuaXgxFDASBgNVBAoTC1B1b3BsZVRvb2xzMRMw⇒
EQYDVQQLLEwpQ
ZW9wbGVzb2Z0MRYwFAYDVQQDEw1NREFXU090MDUxNTAzMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBg⇒
QC43
lCZWxrsyxven5QethAdsLIEEPhhhl7TjA0r8pxp0+ukD8LI7TlTntPOMU535qMGfk/jYtG0QbvpwHDYePy⇒
NMt
Vou6WAs2yr1B+WJSp6Zm42m8PPihfMUXYLG9RiIqcmp2FzdIUi4M07J8ob8rf0W+N11bGW2dmXZ0jGvBm⇒
NHQI
DAQABoAAwDQYJKoZIhvcNAQEEBQADgYEAKx/ugTt0soNVmiH0YcI8FyW8b81FWGIR0f1Cr2MeDi⇒
OQ2pty24dKK
LUqIhogTZdFAN0ed6Ktc82/5xBoH1gv7YeqyPBjvAxw6ekMsg0EzLq90U3ESezZorYFdrQTzqsEXUp1A+c⇒
Zdfo
0eKwZTFmjNash1kis+H0LoQQwyjgaxYI=
-----END NEW CERTIFICATE REQUEST-----
```

The CSR is written in as a text file to the <PS_HOME>\websevr\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.
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 <i>PT845GATEWAY</i> .
Common name for this certificate.	Enter a name, such as <i>PT845GATEWAY</i> .

Setting Up BEA WebLogic for SSL

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.
 - e. Click the Continue button. The Review SSL Private Key Settings page displays.
8. In the Review SSL Private Key Setting page, review the information and click the Continue button.
9. Click the Finish button. You will restart the web server at a later time. You are returned to the Keystore Configuration tab.
10. Scroll down the page to the Advanced Options section and click the Show link.
11. In the Server Attributes section, from the Two Way Client Cert Behavior drop-down list, select *Client Certs Requested and Enforced*.
12. Click the Apply button.
13. Restart the web server.

Installing Digital Certificates for Server Authentication SSL on IBM WebSphere

This section describes how to install digital certificates for server authentication SSL 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 SSL.

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.

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 `<WEBSPPHERE_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
```


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 <i>PT845GATEWAY</i> .
Common name for this certificate.	Enter a name, such as <i>PT845GATEWAY</i> .

Setting Up IBM WebSphere for SSL

Setting up IBM WebSphere for SSL requires that you perform the following tasks:

- Configure SSL repertoires.
- Set up WebSphere servers for SSL.
- 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 Repertoires 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:

1. Open the WebSphere Administration Console, if it is not already open.
The URL is <http://localhost:9090/admin/>.
2. In the left navigation area, select Servers, Application Servers and select the server with which you would like to work. The Application Servers page displays.
3. Click the name of the server that appears as a hyperlink on the page.
4. Click the Configuration tab.
5. In the Additional Properties section, click Web Container. The Web Container page displays.
6. In the Additional Properties section, click the HTTP Transports link.
7. Check the box of the row that contains the entry for the transfer you want to secure.
8. In the Hosts column click the asterisk (*). The HTTP Transports page displays.
9. In the Configuration panel in the General Properties section, for the SSL Enabled property check the Enable SSL box.
10. From the SSL drop-down list, select the desired SSL entry from the repertoire.
11. Click the OK button and save the changes.

To set up CSI authentication:

1. Open the WebSphere Administration Console, if it is not already open.
The URL is <http://localhost:9090/admin/>.
2. In the left navigation area, navigate to Security, AuthenticationProtocol, 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.

Setting Up Digital Certificates for Server Authentication on Application Servers

This section discusses how to set up server-side digital certificates on an application server and describes how to:

- Add CA authorities and root certificates.
- Set up local node certificates.
- Set up remote node certificates.

Adding CA Authorities and 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 add CA authorities and root certificates:

1. Select PeopleTools, Security, Security Objects, Digital Certificates. The Digital Certificates page displays.
2. Add a CA authority:
 - a. Click the plus button (+). A new row appears.
 - b. From the Type drop-down list, select Root CA.
 - c. In the Alias field, enter the alias name for the certificate.
 - d. In the Issuer Alias field, enter an alias for the issuer. Click the Lookup button to select the certificate alias as the issuer alias.
3. Add the root certificate.
4. Click the Add Root link near the plus button (+). The Add Root Certificate page displays.
5. Copy the contents of the certificate into the text box.
 You must include the begin section (-----BEGIN CERTIFICATE-----) and end section (-----END CERTIFICATE -----).
6. Click the OK button.
7. Click the Refresh button.

Setting Up Local Node Certificates

This section discusses setting up local node client certificates and 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 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.
 Do not include any underscores (_) in the node name you enter. For example, if the node name is QE_LOCAL, enter QELOCAL.
 - 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:

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.

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.
 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 node 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.

Setting Up Remote Node Certificates

This section discusses setting up remote node certificates and describes how to:

1. Export remote node certificates.
2. Add remote node CAs and import remote certificates into the local node system.

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.
Do not include any underscores (_) in the node name you enter. For example, if the node name is QE_REMOTE, enter *QEREMOTE*.
 - 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.

Installing Digital Certificates for Client Authentication SSL

This section discusses installing digital certificates for client authentication SSL and discusses how to:

- Generate private and public key pairs.
- Generate CSRs.
- Obtain signed root certificates.
- Import signed root certificates.

Understanding Installing Digital Certificates for Client Authentication SSL

You use the Java-based Keytool command line utility to install gateway-based certificates for SSL with client authentication, which secures outbound messages. 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 is created with the PeopleSoft Pure Internet Architecture installation in the web server directory structure.

The location of Keytool is <PS_HOME>\websrv\<DOMAIN>\keystore\.

You generate a PEM-formatted CSR that contains the gateway's public key. You submit the CSR to the selected CA. The CA creates, digitally signs, and returns your gateway's public key certificate to you. This certificate also contains a signed copy of the CA's root certificate. These certificates may be in standard DER-encoded binary format, or they can be converted to PEM format if necessary.

You then install both signed certificates in the gateway keystore. In addition, you register them and the private key with the web server so that it can recognize and use them.

Understanding the Keytool Utility

You may have previously installed software on the gateway server machine that included a distribution of the Keytool utility. To install client authentication SSL for PeopleSoft Integration Broker, 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>

Generating Private and Public Key Pairs

To generate a key pair:

1. Open a command prompt and navigate to the location of 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:

Option	Specify a name for the key pair; for example, <code>My_GW_Client_Key</code> . You also enter this value in the <code>integrationGateway.properties</code> 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. Use the steps described earlier in this chapter for importing private keys into keystores for the web server you are using to import the keypair 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:

key_alias

Enter the name of the key pair that you created previously; for example:

My_GW_Client_Key

csr_filename

Specify the name of the file that contain the CSR; for example:

My_GW_Client_Key.csr

You can also include a path for the file to create it in a different location than the keystore.

key_password

Enter the password that you specified when you created the key pair.

The CSR file appears in the location and with the name that you specified.

Obtaining Signed Root Certificates

You need to obtain a signed certificate from the selected CA. The signed certificate contains your gateway's public key. The process of obtaining digital certificates varies, depending on the certificate authority that you select. Typically, a CA requires you to paste the content of the PEM-formatted CSR into a form that you submit online. The CA may send you the signed public key certificate by email, or it may require you to download the certificate from a specified page. The CA may also provide its root certificate or instructions for retrieving it. Save the certificates to the location of the keystore file in the following location:

<PS_HOME>\websevr\<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"

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:

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.

Configuring SSL Encryption

Configuring 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 both the web server keystore and the gateway keystore. You configure the web server to recognize and use its installed certificates for SSL transactions.

See [Chapter 17, "Setting Up Secure Messaging Environments," Installing Digital Certificates for Server Authentication SSL on BEA WebLogic, page 401.](#)

See [Chapter 17, "Setting Up Secure Messaging Environments," Installing Digital Certificates for Server Authentication SSL on IBM WebSphere, page 405.](#)

See [Chapter 17, "Setting Up Secure Messaging Environments," Installing Digital Certificates for Client Authentication SSL, page 412.](#)

- Edit the Integration Gateway Certificates section of the integrationGateway.properties file to convey parameters for the 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 4, “Managing Integration Gateways,” Using the integrationGateway.properties File, page 41.](#)

After the appropriate server certificates are installed and configured, the gateway can send and receive secure messages. However, to invoke the secure SSL connection, messages must be directed to an HTTPS URL. To enable outgoing HTTPS, 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.

CHAPTER 18

Tuning Messaging System Performance

This chapter discusses how to:

- Throttle dispatched messages through the messaging system.
- Configure integration gateways for load balancing.

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 Integration Broker Monitor.

See Also

[Chapter 14, “Using Integration Broker Monitor,” Viewing Messaging System Performance Statistics, page 316](#)

Throttling Dispatched Messages Through the Messaging System

You can throttle the number of dispatched messages from a given dispatcher to its associated handler(s).

Throttling the messages that pass through the messaging system enables you to avoid Tuxedo queue saturation due to redundant Tuxedo calls, which result in degraded performance.

You can throttle messages on any of the three pub/sub dispatchers:

- PSBRKDSP
- PSPUBDSP
- PSSUBDSP

To set up dispatcher throttling, you must set the following parameters located in PSADMIN:

- Tuxedo Queue Status Check Count
- Dispatcher List Multiplier
- Dispatcher Channel Max Queue Size

Information for setting these parameters is described earlier in this PeopleBook.

See [Chapter 3, “Administering Messaging Servers for Asynchronous Messaging,” Specifying Dispatcher Parameters, page 28.](#)

Configuring Integration Gateways for Load Balancing

This section discusses how to configure an integration gateways for load balancing.

Understanding Configuring Integration Gateways for Load Balancing

To increase gateway performance you can use load balancing. Load balancing involves the use of a third-party load balancing software product and the installation and configuration of multiple gateways. Then, when messages are sent or published to your messaging system, the load balancing software analyzes the load on installed gateways and determines to which gateway to send the messages to balance the load on all gateways.

For installation and configuration information about your load balancing software, please see the documentation that is included with the product.

Configuring Load Balancing

To configure gateways participating in load balancing, you must specify the URLs of the gateways in use for load balancing on the Gateways page, and then set integration gateway properties for each gateway you specify. Note that you can set different properties for each gateway.

To access the Gateways page, select PeopleTools, Integration Broker, Gateways. Select the default local gateway.

Gateways

Gateway ID LOCAL

☒ Local Gateway ☒ Load Balancer

URL

Physical Gateways
[Customize](#) | [Find](#) | [View All](#) | [First](#) | [1 of 1](#) | [Last](#)

URL	Properties
1	Properties + -

Connectors
[Customize](#) | [Find](#) | [First](#) | [1-10 of 10](#) | [Last](#)

*Connector ID	Description	*Connector Class Name	Properties	+	-
1 FILEOUTPUT		SimpleFileTargetConnector	Properties	+	-
2 FTPTARGET		FTPTargetConnector	Properties	+	-
3 GETMAILTARGET		GetMailTargetConnector	Properties	+	-
4 HTTPTARGET		HttpTargetConnector	Properties	+	-
5 JMSTARGET		JMSTargetConnector	Properties	+	-
6 LDAPTARGET		LDAPTargetConnector	Properties	+	-
7 POP3TARGET		POP3TargetConnector	Properties	+	-
8 PSFT81TARGET		ApplicationMessagingTargetConnector	Properties	+	-
9 PSFTTARGET		PeopleSoftTargetConnector	Properties	+	-
10 SMTPTARGET		SMTPTargetConnector	Properties	+	-

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 4, “Managing Integration Gateways,” page 35](#)

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.
- Set up inbound and outbound transactions.

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.

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 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 Node Definitions component (IB_NODE) to configure local and remote nodes.

Note. If you upgraded from a PeopleSoft 8.1x system, all application messaging nodes that are defined for that system have been preserved as remote node definitions in PeopleSoft Integration Broker. You must assign each of them an appropriate target connector.

When you configure nodes, use the Node Definitions - Connectors page to specify which gateway and target connector should be used 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:

Scenario	System	Node Definition	Connector
PeopleSoft Integration Broker to PeopleSoft Integration Broker	Both applications	Default local	Not applicable (NA)*
PeopleSoft Integration Broker to PeopleSoft Integration Broker	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*

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) • POP3TARGET (Post Office Protocol 3 [POP3] 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 • POP3TARGET
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*

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 4, “Managing Integration Gateways,” Administering Integration Gateways, page 36](#)

[Chapter 9, “Configuring Nodes and Transactions,” Configuring Nodes, page 196](#)

[Chapter 9, “Configuring Nodes and Transactions,” Configuring Transactions, page 204](#)

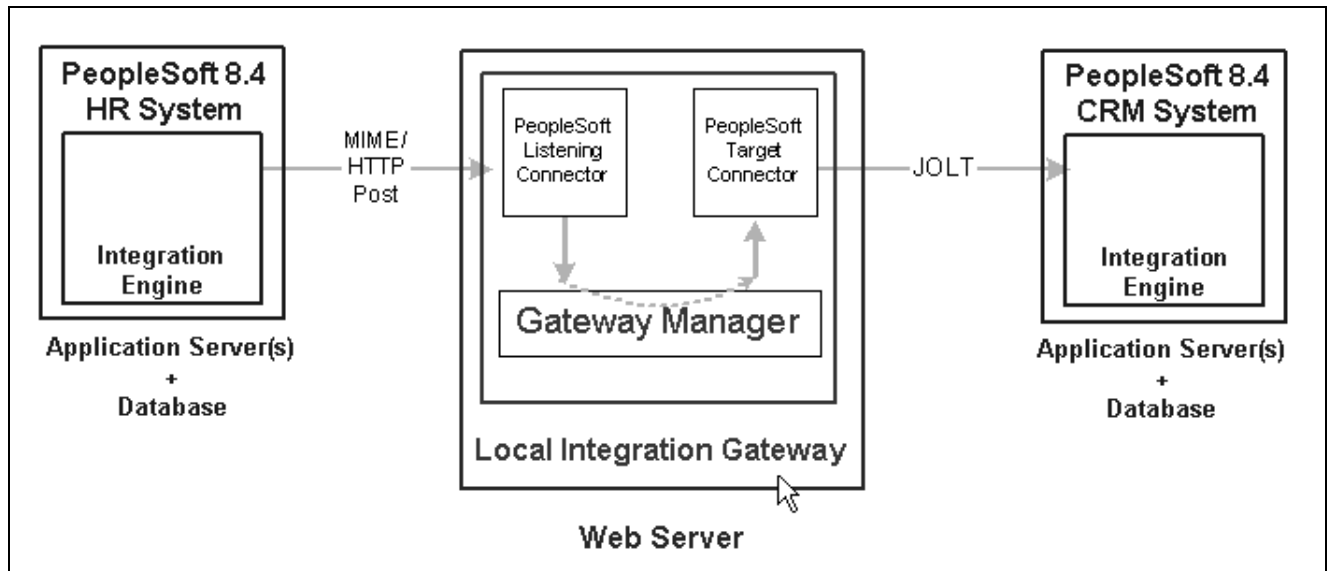
[Chapter 13, “Administering Relationships,” page 273](#)

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:



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 Node Definitions component.

This node is delivered predefined on the system.

3. Define a remote node to represent the PeopleSoft CRM system.

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 outbound transactions as part of the PeopleSoft CRM node definition.

These transactions route messages from the PeopleSoft Human Resources system to its local gateway through the gateway's PeopleSoft listening connector. The gateway uses its PeopleSoft target connector to send the messages to the PeopleSoft CRM system.

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 Node Definitions 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 inbound transactions as part of the PeopleSoft Human Resources node definition.

When the PeopleSoft Human Resources system sends messages to its local gateway, the gateway uses these transactions to deliver the messages to the PeopleSoft CRM system by using its PeopleSoft target connector.

See Also

[Appendix A, "Integration Scenarios," Understanding Integration Setup, page 421](#)

[Chapter 4, "Managing Integration Gateways," Administering Integration Gateways, page 36](#)

[Chapter 9, "Configuring Nodes and Transactions," Configuring Nodes, page 196](#)

[Chapter 9, "Configuring Nodes and Transactions," Configuring Transactions, page 204](#)

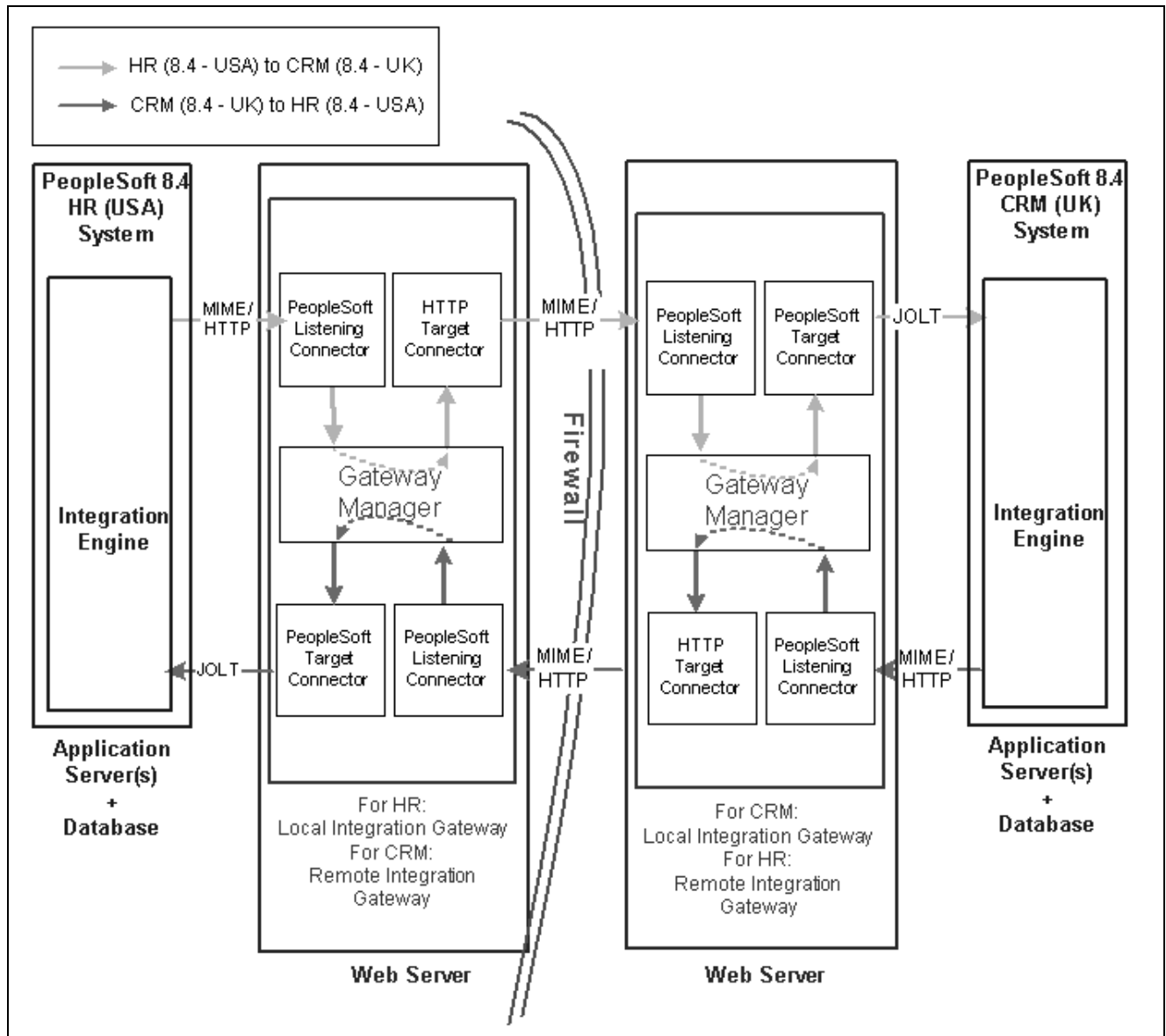
Integrating with PeopleSoft Integration Broker Systems Through a Firewall

This section provides an overview of this scenario and discusses how to configure the system for this scenario.

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, 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.

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:

- Define a local integration gateway.

Use the Gateways component to define the local PeopleSoft Human Resources (USA) gateway.

- 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.

- Define the default local node.

Use the Node Definitions component to define the default local node, which represents the PeopleSoft Human Resources (USA) system.

- 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.

- Define outbound transactions.

Use the Node Definitions - Transactions page to define outbound transactions to the PeopleSoft CRM (UK) system on the PeopleSoft CRM (remote) node definition.

- Define inbound transactions.

Use the Node Definitions - Transactions page to define inbound transactions from the PeopleSoft CRM (UK) system on the PeopleSoft CRM (remote) node definition.

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:

- Define a local integration gateway.

Use the Gateways component to define the local PeopleSoft CRM (UK) gateway.

- 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.

- Define the default local node.

Use the Node Definitions component to define the default local node, which represents the PeopleSoft CRM (UK) system.

- 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.

- Define outbound transactions.

Use the Node Definitions - Transactions page to define outbound transactions to the PeopleSoft Human Resources (USA) system on the PeopleSoft Human Resources (remote) node definition.

- Define inbound transactions.

Use the Node Definitions - Transactions page to define inbound transactions from the PeopleSoft Human Resources (USA) system on the PeopleSoft Human Resources (remote) node definition.

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 421](#)

[Chapter 4, “Managing Integration Gateways,” Administering Integration Gateways, page 36](#)

[Chapter 9, “Configuring Nodes and Transactions,” Configuring Nodes, page 196](#)

[Chapter 9, “Configuring Nodes and Transactions,” Configuring Transactions, page 204](#)

Integrating with PeopleSoft Integration Broker Systems by Using Hubs

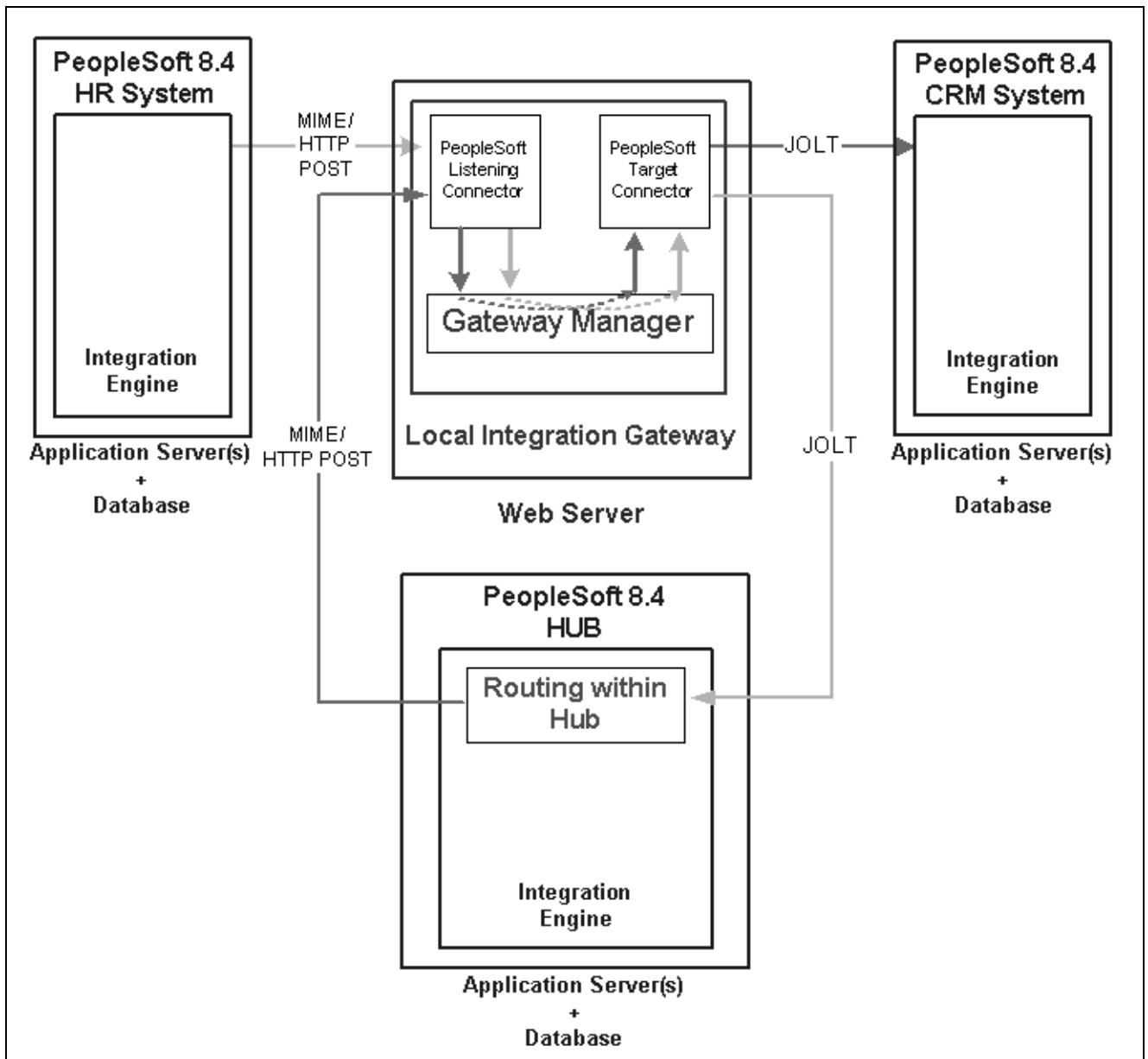
This section provides overviews of this scenario and hub routing types and discusses how to configure:

- Generic routing hubs.
- Sender-specified routing hubs.

Understanding This Scenario

A PeopleSoft Integration Broker hub configuration includes an integration engine that houses routing rules and transformations. All transactions are routed through the hub, which enables you to centralize routing rules and offload the transformation process.

This diagram shows a 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.

With sender-specified routing, a destination node name is passed as a parameter to a PeopleCode Publish or SyncRequest method, such as PublishXMLDoc or SyncRequestXMLDoc, 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:

- Define a local integration gateway.
Use the Gateways component to set up a local integration gateway for sending messages.
- Set up a local node.
Use the Node Definitions component to set up the local node, which represents the PeopleSoft Human Resources system.
- Set up a remote node.
Use the Node Definitions component to set up the remote node, which represents the hub system.
- Set up outbound transactions.
Use the Node Definitions - Transactions page to set up outbound transactions on the hub (remote) node.

Configuring the PeopleSoft CRM System

On the PeopleSoft CRM system:

- Define a local integration gateway.
Use the Gateways component to set up a local integration gateway for sending messages.
- Set up a local node.
Use the Node Definition component to set up the local node, which represents the PeopleSoft CRM system.
- Set up a remote node.
Use the Node Definition component to set up the remote node, which represents the hub system.
- Set up inbound transactions.
Use the Node Definitions - Transactions page to set up inbound transactions from the hub node.

Configuring the PeopleSoft Hub

On the PeopleSoft hub:

- Define a local integration gateway.
Use the Gateways component to set up a local integration gateway for sending messages.
- Set up a local node.
Use the Node Definition component to set up the local node, which represents the hub system.
- 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.
- Set up inbound transactions.
Use the Node Definitions - Transactions page to create inbound transactions from the PeopleSoft Human Resources system.
- Set up outbound transactions.
Use the Node Definitions - Transactions page to create outbound transactions for the PeopleSoft CRM system.
- Set up relationships.
Use the Relationships component (IB_RELATIONSHIP) to set up relationships to link the inbound transactions from the PeopleSoft Human Resources system to the outbound transactions for 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 transactions and relationships on the hub. By using the systems in the diagram as an example, the following table shows the node, transaction, and relationship configurations that are required for generic routing through a hub:

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.
Transactions	Define outbound transaction to the hub system.	Define inbound transaction from the PeopleSoft Human Resources system. Define outbound transaction to the PeopleSoft CRM system.	Define inbound transaction from the hub system.
Relationship or transaction modifier	NA	Define modifier to convert inbound from PeopleSoft Human Resources to outbound for PeopleSoft CRM.	NA

See Also

[Appendix A, “Integration Scenarios,” Understanding Integration Setup, page 421](#)

[Chapter 4, “Managing Integration Gateways,” Administering Integration Gateways, page 36](#)

[Chapter 9, “Configuring Nodes and Transactions,” Configuring Nodes, page 196](#)

[Chapter 9, “Configuring Nodes and Transactions,” Configuring Transactions, page 204](#)

[Chapter 13, “Administering Relationships,” page 273](#)

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.

Configuring PeopleSoft Human Resources (Sending) System

On the PeopleSoft Human Resources system:

- Define a local integration gateway.

Use the Gateways component to set up a local integration gateway for sending messages.

- Set up a local node.

Use the Node Definitions component to set up the local node, which represents the PeopleSoft Human Resources system.

- 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, you must set the routing type to Explicit and for the hub node, you must enter the node name of the hub.

- Set up outbound transactions.

Use the Node Definitions - Transactions page to set up outbound transactions on the PeopleSoft CRM system.

Configuring PeopleSoft CRM (Receiving System)

On the PeopleSoft CRM system:

- Define a local integration gateway.

Use the Gateways component to set up a local integration gateway for sending messages.

- Set up a local node.

Use the Node Definitions component to set up the local node, which represents the PeopleSoft CRM system.

- Set up a remote node.

Use the Node Definitions component to set up a node that represents the hub.

- Set up inbound transactions.

Use the Node Definitions - Transactions page to create inbound transactions from the hub.

Configuring the PeopleSoft Hub

On the PeopleSoft hub:

- Define a local integration gateway.

Use the Gateways component to set up a local integration gateway for sending messages.

- Set up a local node.

Use the Node Definitions component to set up the local node, which represents the hub system.

- Set up remote nodes.

Use the Node Definitions component to set up two remote nodes: one for the PeopleSoft Human Resources system and one for the PeopleSoft CRM system.

- Set up inbound transactions.

Use the Node Definitions - Transactions page to create inbound transactions from the PeopleSoft Human Resources system.

- Set up outbound transactions.

Use the Node Definitions - Transactions page to create outbound transactions for the PeopleSoft CRM system.

- Set up relationships.

Use the Relationships component to set up relationships for the PeopleSoft CRM system. These relationships links inbound transactions from the PeopleSoft Human Resources system to the PeopleSoft CRM system.

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.
Transactions	Define outbound transaction to PeopleSoft CRM.	Define inbound transaction from the PeopleSoft Human Resources system. Define outbound transaction to the PeopleSoft CRM system.	Define inbound transaction from the hub.
Relationship or transaction modifier	NA	Define a modifier to convert inbound from PeopleSoft Human Resources to outbound for PeopleSoft CRM.	NA

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)
- PublishXMLDoc(&MyDoc, Message.MyMessage, Node.CRM)
- SyncRequestXMLDoc(&MyDoc, Message.MyMessage, Node.CRM)

See Also

[Appendix A, “Integration Scenarios,” Understanding Integration Setup, page 421](#)

[Chapter 4, “Managing Integration Gateways,” Administering Integration Gateways, page 36](#)

[Chapter 9, “Configuring Nodes and Transactions,” Configuring Nodes, page 196](#)

[Chapter 9, “Configuring Nodes and Transactions,” Configuring Transactions, page 204](#)

[Chapter 13, “Administering Relationships,” page 273](#)

Integrating with Third-Party Systems

This section provides an overview of this scenario and discusses how to configure the system for this scenario.

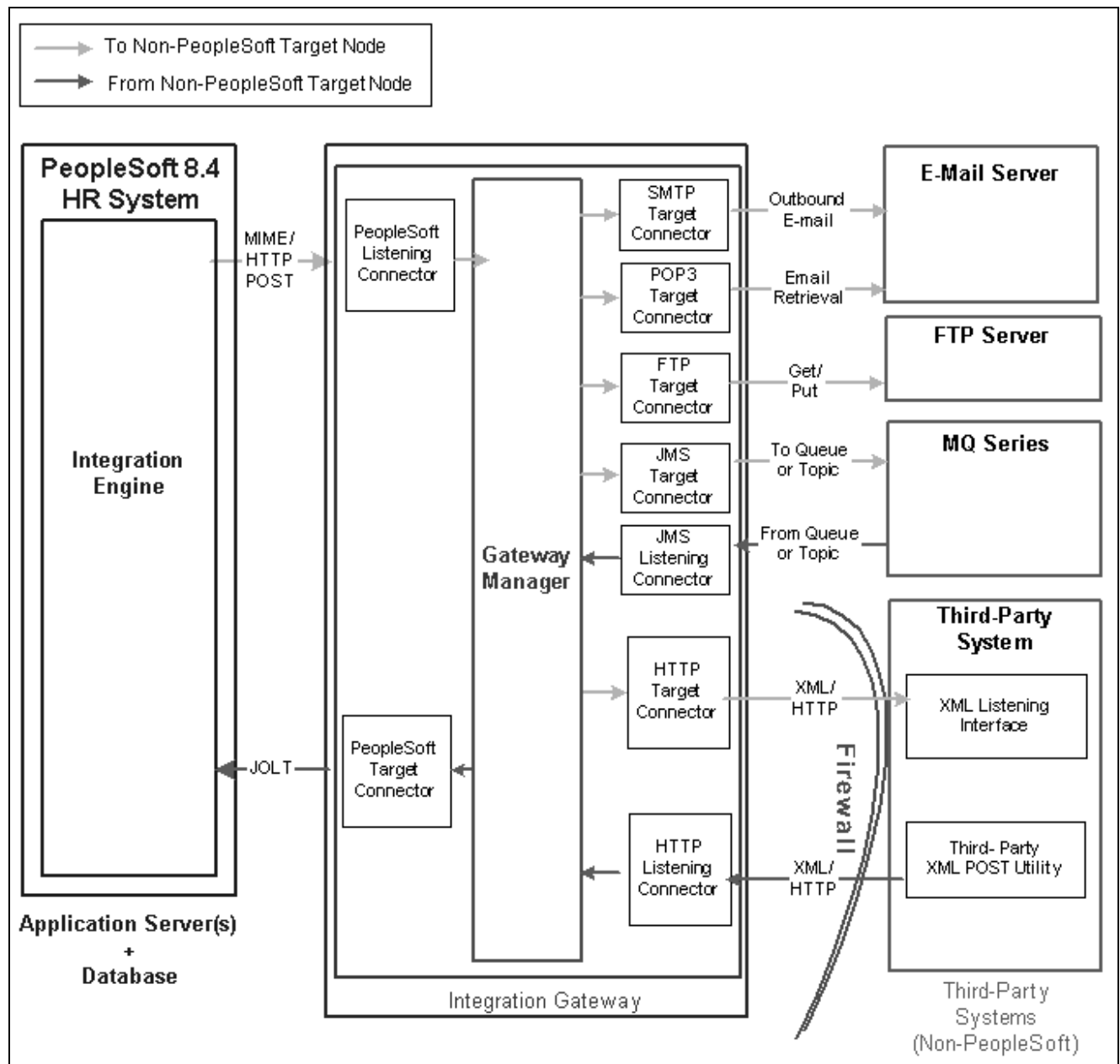
Understanding This Scenario

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:



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:

- Define a local integration gateway.

Use the Gateways component to set up a local integration gateway for sending messages.

- Set up a local node.

Use the Node Definitions component to set up the local node, which represents the PeopleSoft Human Resources system.

- Set up a remote node.

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, POP3 target connector, SMTP target connector, and so forth) for communicating with the third-party system.

- Set up inbound transactions.

Set up inbound transactions on the third-party (remote) node.

- Set up outbound transactions.

Set up outbound transactions on the third-party (remote) node.

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 421](#)

[Chapter 4, “Managing Integration Gateways,” Administering Integration Gateways, page 36](#)

[Chapter 9, “Configuring Nodes and Transactions,” Configuring Nodes, page 196](#)

[Chapter 9, “Configuring Nodes and Transactions,” Configuring Transactions, page 204](#)

[Chapter 13, “Administering Relationships,” page 273](#)

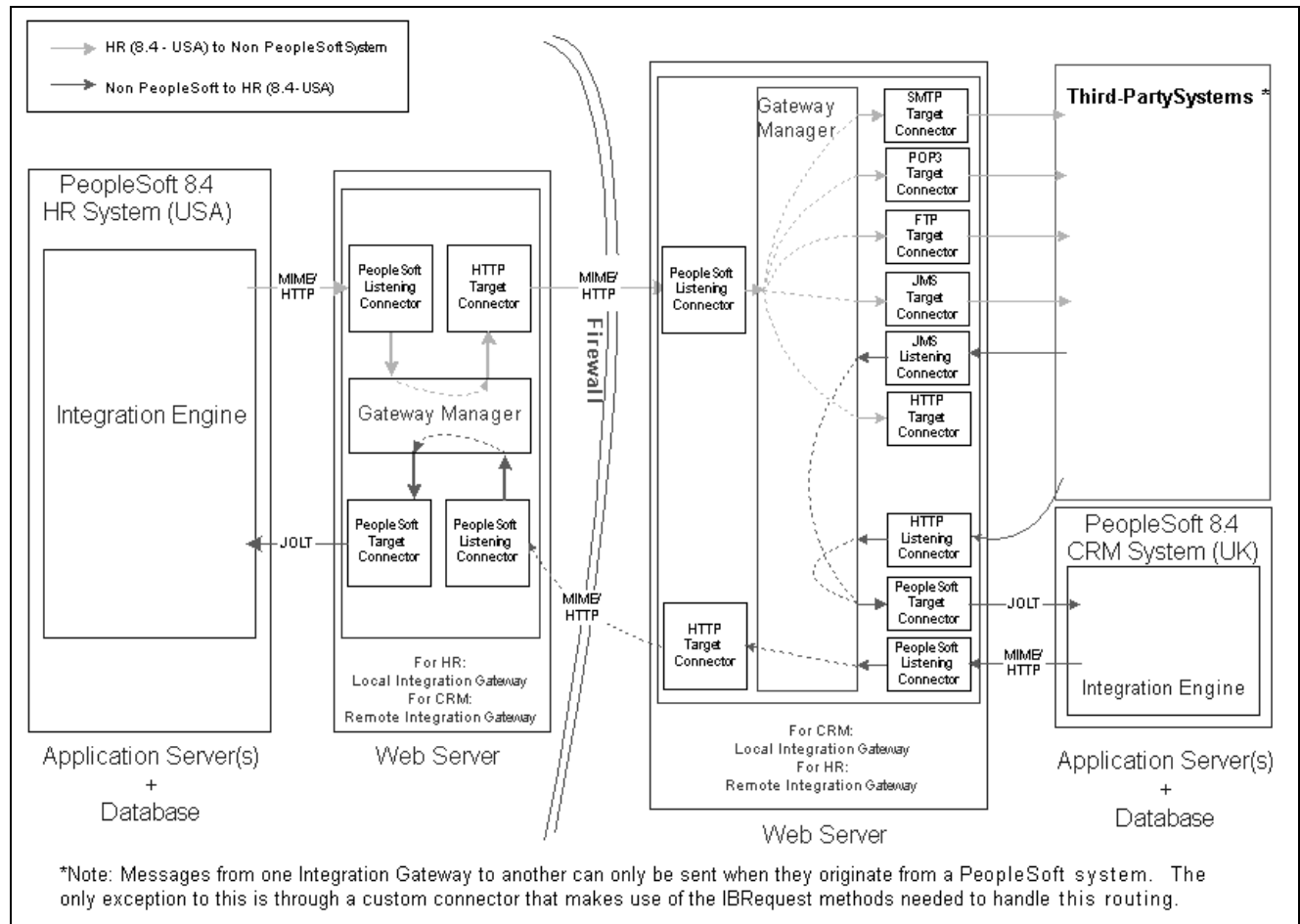
Integrating with Third-Party Systems by Using Remote Gateways

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:



Integrations with third-party systems using remote gateways

Sending Messages to Third Party Systems

The set up for this scenario is similar to the configuration described in the scenario 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 as an example, 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.

PeopleSoft Human Resource System Configuration Tasks

On the PeopleSoft Human Resources system:

- *Define a local integration gateway.* Use the Gateways component to define the local PeopleSoft Human Resources (USA) gateway.

- *Define a remote integration gateway.* The remote gateway for the PeopleSoft Human Resources (USA) system is the CRM (UK) gateway. Use the Gateways component to define the gateway, and specify the URL of the CRM (UK) gateway.
- *Set up a local node.* The local node you set up represents the PeopleSoft Human Resources (USA) system. Use the Node Definitions component to perform this task.
- *Set up a remote node.* The remote node you set up represents the third-party system. Use the Node Definitions component to perform this task. When you define the remote node, use the Connectors tab 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.
- *Set up inbound transactions.* Set up inbound transactions on the third-party (remote) node. This is only required if you will be receiving messages from this third-party system.
- *Set up outbound transactions.* Set up outbound transactions on the third-party (remote) node.

PeopleSoft Human Resources Integration Gateway Configuration Tasks

The only required integration gateway properties are the Jolt connect string properties that enable communication with integration engines on Integration Broker systems. Use the `integrationGateway.properties` file to set these properties.

See Also

[Appendix A, “Integration Scenarios,” Understanding Integration Setup, page 421](#)

[Chapter 4, “Managing Integration Gateways,” Administering Integration Gateways, page 36](#)

[Chapter 9, “Configuring Nodes and Transactions,” Configuring Nodes, page 196](#)

[Chapter 9, “Configuring Nodes and Transactions,” Configuring Transactions, page 204](#)

[Chapter 13, “Administering Relationships,” page 273](#)

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.

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.

PeopleSoft Human Resources (USA) System Configuration Tasks

On the PeopleSoft Human Resources system:

- *Set up a local node.* The local node you set up represents the PeopleSoft Human Resources system. Use the Node Definition component to perform this task.
- *Set up a remote node.* The remote node you set up represents the CRM system. When you set up the remote node, specify the PeopleSoft target connector.
- *Set up inbound transactions.* Set up inbound transactions on the CRM (remote) node.
- *Set up outbound transactions.* Set up outbound transactions on the CRM (remote) node.

PeopleSoft Human Resources Integration Gateway Configuration Tasks

The only required integration gateway properties are the Jolt connect string properties that enable communication with the PeopleSoft Human Resources system. Use the `integrationGateway.properties` file to set these properties.

PeopleSoft CRM (UK) System/Hub Configuration Tasks

In this scenario, the PeopleSoft CRM (UK) system serves as a hub. On the PeopleSoft CRM (UK) system:

- *Define a local integration gateway.* Using the Gateways Component 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.* The local node you set up represents the CRM system. Use the Node Definition component to perform this task.

- *Set up remote nodes.* Define two remote nodes; define one remote node that represents the third-party system and define another to represent the PeopleSoft Human Resources/USA system. Use the Node Definition component to perform this task.

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 gateway (on the USA gateway).

- *Set up inbound transactions.* Set up inbound transaction on the third-party (remote) node. See the table at the end of this section for information about setting up transactions for this scenario.
- *Set up outbound transactions.* Set up outbound transactions on the PeopleSoft Human Resources (remote) node. See the table at the end of this section for information about setting up transactions for this scenario.
- *Set up outbound relationships.* These relationships link inbound transactions from the third-party system to outbound transactions to the PeopleSoft Human Resources system. Use the Relationships component to perform this task.

PeopleSoft Integration Broker CRM (UK) Integration Gateway Configuration Tasks

The only required properties are the Jolt connect string properties that enable communication with integration engines on the PeopleSoft CRM systems. Use the integrationGateway.properties file to set these properties.

Third Party System to PeopleSoft System Configuration Summary

Since 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 get routed through it. Using the systems in the diagram as an example, the following table shows the required node, transaction and relationship configurations.

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 CRM system.
Remote Nodes	Define a remote node to represent the CRM system.	Define remote nodes to represent the third-party system and the PeopleSoft Human Resources system.
Transactions	Define an inbound transaction from the CRM system.	Define an inbound transaction from the third-party system. Define an outbound transaction to the PeopleSoft Human Resources system.
Relationship/Transaction Modifier	N/A	Define modifier to convert inbound from third party to outbound for the PeopleSoft Human Resources system.

See Also

[Appendix A, “Integration Scenarios,” Understanding Integration Setup, page 421](#)

[Chapter 4, “Managing Integration Gateways,” Administering Integration Gateways, page 36](#)

[Chapter 9, “Configuring Nodes and Transactions,” Configuring Nodes, page 196](#)

[Chapter 9, “Configuring Nodes and Transactions,” Configuring Transactions, page 204](#)

[Chapter 13, “Administering Relationships,” page 273](#)

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.

Configuring the PeopleSoft Human Resources System

On the PeopleSoft Human Resources system:

- Define a local integration gateway.
Use the Gateways component to define the local PeopleSoft Human Resources (USA) gateway.
- Define a remote integration gateway.
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.
- Set up a local node.
Use the Node Definitions component to set up the local node, which represents the PeopleSoft Human Resources (USA) system.
- Set up a remote node.
Use the Node Definitions 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.
- Set up inbound transactions.

Set up inbound transactions on the third-party (remote) node. This is required only if you will be receiving messages from this third-party system.

- Set up outbound transactions.

Set up outbound transactions on the third-party (remote) node.

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 421](#)

[Chapter 4, “Managing Integration Gateways,” Administering Integration Gateways, page 36](#)

[Chapter 9, “Configuring Nodes and Transactions,” Configuring Nodes, page 196](#)

[Chapter 9, “Configuring Nodes and Transactions,” Configuring Transactions, page 204](#)

[Chapter 13, “Administering Relationships,” page 273](#)

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.
- 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:

- Set up a local node.
Use the Node Definition component to set up the local node, which represents the Human Resources system.
- 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.
- Set up inbound transactions.
Set up inbound transactions on the PeopleSoft CRM (remote) node.
- Set up outbound transactions.
Set up outbound transactions on the PeopleSoft CRM (remote) node.

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.

- Set up inbound transactions.

Set up inbound transaction on the third-party (remote) node according to the table at the end of this section.

- Set up outbound transactions.

Set up outbound transactions on the PeopleSoft Human Resources (remote) node according to the table at the end of this section.

- Set up outbound relationships.

Use the Relationships component to set up relationships, which link inbound transactions from the third-party system to outbound transactions to the PeopleSoft Human Resources system.

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:

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.

Item to Configure	PeopleSoft Human Resources System	PeopleSoft CRM System (Hub)
Transactions	Define an inbound transaction from the PeopleSoft CRM system.	Define an inbound transaction from the third-party system. Define an outbound transaction to the PeopleSoft Human Resources system.
Relationship or transaction modifier	NA	Define a modifier to convert inbound from third-party to outbound for PeopleSoft Human Resources.

See Also

[Appendix A, “Integration Scenarios,” Understanding Integration Setup, page 421](#)

[Chapter 4, “Managing Integration Gateways,” Administering Integration Gateways, page 36](#)

[Chapter 9, “Configuring Nodes and Transactions,” Configuring Nodes, page 196](#)

[Chapter 9, “Configuring Nodes and Transactions,” Configuring Transactions, page 204](#)

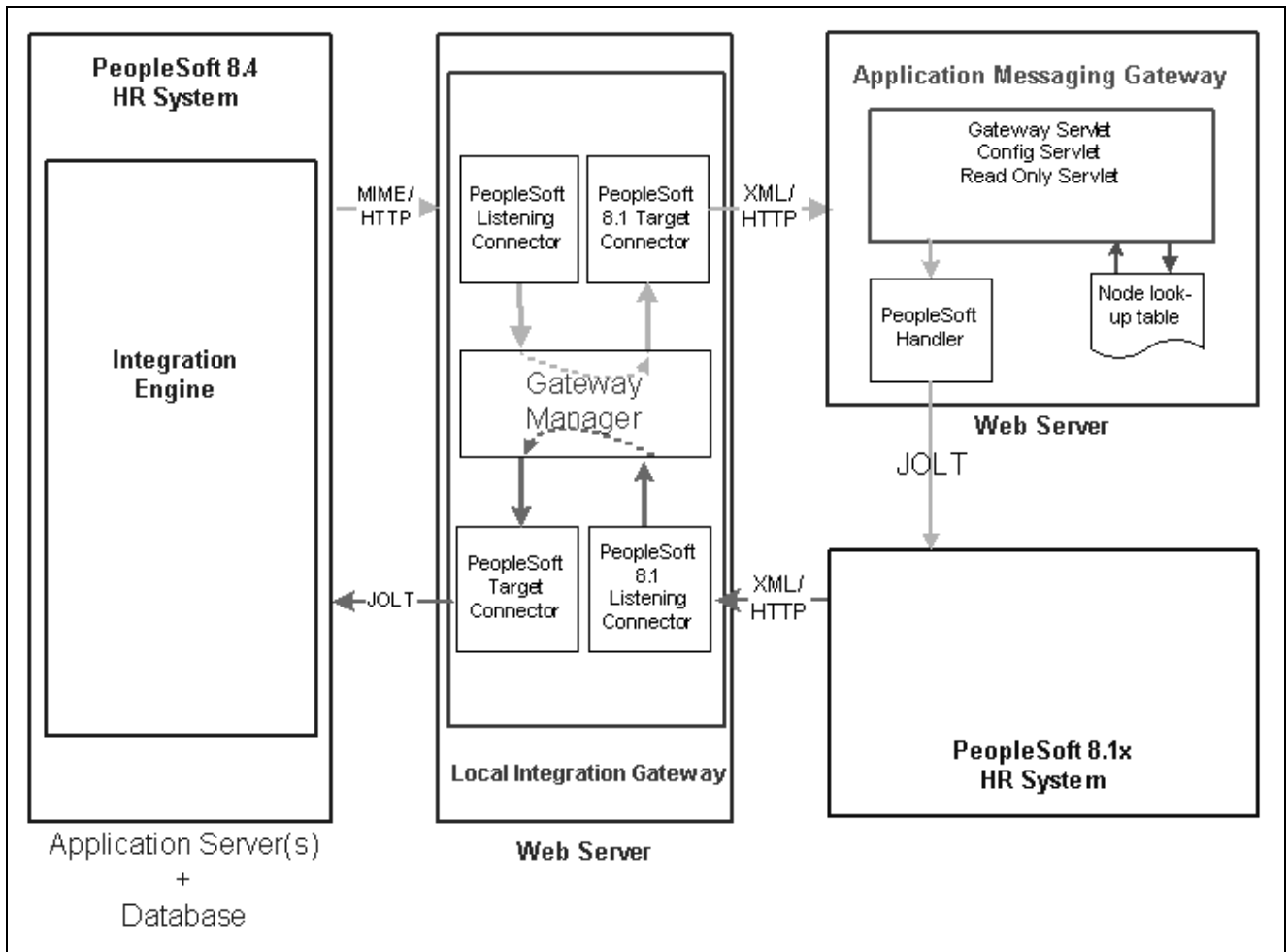
[Chapter 13, “Administering Relationships,” page 273](#)

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:



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.

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.

- Set up inbound transactions.

Set up inbound transactions on the PeopleSoft 8.1x Human Resources (remote) node.

- Set up outbound transactions.

Set up outbound transactions on the 8.1x Human Resources (remote) node.

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 421](#)

[Chapter 4, “Managing Integration Gateways,” Administering Integration Gateways, page 36](#)

[Chapter 9, “Configuring Nodes and Transactions,” Configuring Nodes, page 196](#)

[Chapter 9, “Configuring Nodes and Transactions,” Configuring Transactions, page 204](#)

[Chapter 13, “Administering Relationships,” page 273](#)

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 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.

Note. While still delivered with PeopleSoft Integration Broker, the POP3 target connector is a deprecated technology. Its functionality is now provided with the GetMail target connector. For an example of how to use the GetMail connector, please refer to the PeopleSoft Multichannel Framework PeopleBook.

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 example and choose to cut the code examples 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 examples to fail.

See Also

Enterprise PeopleTools 8.45 PeopleBook: PeopleSoft MultiChannel Framework, "Configuring the Email Channel"

Prerequisites

To use this appendix, you should have basic experience using PeopleSoft Integration Broker. There is little background information presented in this appendix and many of the basic steps involved in creating integrations are presented in general terms (for example, “create a new message.”) Please refer to the appropriate chapters in this PeopleBook for information on how to complete basic tasks.

Setting Up Meta Data

This section discusses how to set up meta data for the examples presented in this appendix and discusses how to:

- Create message channels, request messages, response messages, and pages.
- Create nodes and transactions.
- Set up integration gateway logging.

Understanding Setting Up Meta Data

Before you use the examples in this appendix, you must set up meta data as described in this section.

Note. The examples presented in this appendix demonstrate using one type of connector at a time. The examples share the same basic definitions for the request message, response message, transactions, and the test page. As a result, you should attempt to run only one example at a time, since the underlying meta data 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 Message Channels, Messages, and Pages

This section describes creating message channels, request and response messages, and pages for use in the running the connector examples presented in this appendix.

Unless otherwise notes, use PeopleSoft Application Designer to complete these tasks.

To create message channels, messages, and pages:

1. Create a new message channel.
 - a. Name the message channel *EXAMPLE_CHANNEL*
 - b. Verify in the channel properties that the Status is set to Run.
 - c. In the PeopleSoft Pure Internet Architecture, configure the security for the Integration Broker Monitor so that the channel can be displayed.
 - d. In the PeopleSoft Pure Internet Architecture, use the Integration Broker Monitor to verify that the *EXAMPLE_CHANNEL* is running.
2. Create a new request message.

- a. Open the properties and select the Active box for the Status.
 - b. Set the message channel to *EXAMPLE_CHANNEL*.
 - c. Save this message as *EXAMPLE_REQUEST_MSG*.
3. Create a new response message.
 - a. Open the properties and select the Active box for the Status.
 - b. Set the message channel to *EXAMPLE_CHANNEL*.
 - c. Save this message as *EXAMPLE_RESPONSE_MSG*.
4. Create a new record.
 - a. Insert the character field *TEST* into the record.
 - b. Select Derived/Work as the Record Type.
 - c. Save the record as *EXAMPLE_WORKREC*.
5. Create a new page with a single push button.
 - a. Create a new page.
 - b. Add a push button with the following properties:

Property	Value
Destination	PeopleCode Command
RecordName	<i>EXAMPLE_WORKREC</i>
Field Name	<i>TEST</i>

- c. Resize the button and label it *Test target connector*.
- d. Save the page as *EXAMPLE_PAGE*.
- e. 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 Transactions

Use the PeopleSoft Pure Internet Architecture to complete the following tasks.

Creating Source Nodes and Transactions

You must create a node that will be the source of all requests to the listening connectors.

To create a source node and a transaction:

1. Add a new node called *SOURCENODE*. Verify that the Active Node check box has been selected. Save this node.
2. Add a new transaction to the *SOURCENODE*.
 - a. Complete the field definitions as follows:

Field	Value
Transaction Type	Inbound Synchronous
Request Message	<i>EXAMPLE_REQUEST_MSG</i>
Message Version	<i>VERSION_1</i>

- b. On the Transactions tab, verify that the Status is set to Active.
- c. On the Messages tab, verify that Synchronous Logging is set to Header and Detail.
- d. Add the response message details as follows:

Field	Value
Message Name	<i>EXAMPLE_RESPONSE_MSG</i>
Message Version	<i>VERSION_1</i>

- e. Save the transaction.

Adding Target Nodes and Transactions

You must create a target node and a transaction for all outgoing requests for the target connectors.

To add a target node and a transaction:

1. Add a new node called *TARGETNODE*. Verify that the Active Node check box has been selected. Save this node.
2. Add a new transaction to the *TARGETNODE*.
 - a. Complete the field definitions as follows:

Field	Value
Transaction Type	Outbound Synchronous
Request Message	<i>EXAMPLE_REQUEST_MSG</i>
Message Version	<i>VERSION_1</i>

- b. On the Transactions tab, verify that the Status is set to Active.
- c. On the Messages tab, verify that Synchronous Logging is set to Header and Detail.
- d. Add the response message details as follows:

Field	Value
Message Name	<i>EXAMPLE_RESPONSE_MSG</i>
Message Version	<i>VERSION_1</i>

- e. Save the transaction.
3. Add a new transaction to the *TARGETNODE*.
 - a. Complete the field definitions as follows:

Field	Value
Transaction Type	Outbound Asynchronous
Request Message	<i>EXAMPLE_REQUEST_MSG</i>
Message Version	<i>VERSION_1</i>

- b. On the Transactions tab, verify that the Status is set to Active.
- c. Save the transaction.

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 15, “Managing Error Handling, Logging, Tracing, and Debugging,” Managing Integration Gateway Message and Error Logging, page 353.](#)

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 send an asynchronous message between two PeopleSoft nodes.

The example provided for using the PeopleSoft listening connector demonstrates using Send Master to send a message into the local system for processing.

Prerequisites

To use the PeopleSoft target connector example you must have a second PeopleSoft 8.4x system. You must have the application server, the PeopleSoft Pure Internet Architecture and the Integration Gateway configured and running.

Note. In this section, the current PeopleSoft system is referred to as the *originating* system, and the second PeopleSoft system is called the *destination* system.

Using the PeopleSoft Target Connector

This section provides an example of using the PeopleSoft target connector and describes how to:

- Set up data on the originating system.
- Set up data on the destination system.
- Test the PeopleSoft target connector.

Setting Up Data on the Originating System

To set up data on the originating system:

1. In PeopleSoft Application Designer, create a new request message.
 - a. Open the properties and select the Active box for the Status.
 - b. Set the message channel to *EXAMPLE_CHANNEL*.
 - c. Save this message as *EXAMPLE_PSFT_MSG*.
2. Open the *EXAMPLE_WORKREC* record.
 - a. 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);

/* and send it out in an async request */
&success = PublishXmlDoc(&xmlDoc, Message.EXAMPLE_PSFT_MSG);

MessageBox(0, "", 0, 0, "Message sent.");
```
 - b. Save the changes
3. In the PeopleSoft Pure Internet Architecture, use the Node Definitions component to add a new node.
 - a. Name the node using the name of the destination system.
 - b. Verify that the Active Node box is selected.
 - c. Save the changes.
4. Use the Node Definitions — Transaction tab to add an outbound transaction to the new node.
 - a. Complete the field definitions as follows:

Field	Value
Transaction Type	Outbound Asynchronous
Request Message	<i>EXAMPLE_PSFT_MSG</i>
Request Message Version	<i>VERSION_1</i>

- b. In the Transactions Detail section, verify that the Status is set to Active.
 - c. Save the transaction.
5. On the Node Definitions-Connectors page, set the Connector ID to *PSFTTARGET* and save the changes.
6. Use the Gateways component to set the BEA Jolt connection information for the destination node.

Setting Up Data on the Destination System

To set up data on the destination system:

1. Create a new message channel called *EXAMPLE_CHANNEL*.
 - a. Verify in the channel properties that the Status is set to Run.

- b. Configure the security for the Integration Broker Monitor so that the channel can be displayed.
 - c. In Integration Broker Monitor, verify that the *EXAMPLE_CHANNEL* is running.
2. Create a new request message.
 - a. Open the properties and select the Active box for the Status.
 - b. Set the message channel to *EXAMPLE_CHANNEL*.
 - c. Save this message as *EXAMPLE_PSFT_MSG*.
3. Add a subscription to the *EXAMPLE_PSFT_MSG* message.
 - a. Name the subscription *ExampleSubscription*.
 - b. Add the following PeopleCode to the subscription body.


```
/* get the incoming message */
&msg = GetMessage();
&msgXML = &msg.GenXMLString();

/* and write it to a file */
&file = GetFile("PSFTmsg.txt", "w");
&file.writeString(&msgXML);
&file.close();
```
4. In the PeopleSoft Pure Internet Architecture, use the Node Definitions component to add a node.
 - a. Name the node using the name of the originating system.
 - b. Verify that the Active Node box is selected.
 - c. Save the changes.
5. Use the Node Definitions — Transaction tab to add an inbound transaction to the new node.
 - a. Complete the field definitions as follows:

Field	Value
Transaction Type	Inbound Asynchronous
Request Message	<i>EXAMPLE_PSFT_MSG</i>
Request Message Version	<i>VERSION_1</i>

- b. In the Transactions Detail section, verify that the Status is set to Active.
 - c. Save the transaction.

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.
2. On the originating system, open Integration Broker Monitor to view the details of the message sent to the destination system.
3. On the destination system, open Integration Broker Monitor to view the details of the received message. Open the *PSFTmsg.txt* file created by the subscription PeopleCode to view the body of the message.

Using the PeopleSoft Listening Connector

This section provides an example for testing the PeopleSoft listening connector.

Testing the PeopleSoft Listening Connector

To test the PeopleSoft listening connector:

1. In PeopleSoft Application Designer, open the `EXAMPLE_REQUEST_MSG` and add the following PeopleCode to the `OnRequest` message event.

```
/* get the body of the incoming message */
&xmlDoc = GetMessageXmlDoc();

/* and write it out to a file */
&theFile = GetFile("PSFTmsg.txt", "w");
&theFile.writestring(&xmlDoc.GenXmlString());
&theFile.Close();

/* create the response message */
&msg = CreateMessage(Message.EXAMPLE_RESPONSE_MSG);

/* 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 */
&msg.SetXmlDoc(&xmlDoc);

/* and send the response message */
ReturnToServer(&msg);
```

2. Start Send Master and create an 8.4 Integration Broker (MIME) project.
3. In the URL field enter the address of the PeopleSoft listening connector:

```
http://your_server_name/PSIGW/PeopleSoftListeningConnector
```

Replacing `<your_server_name>` with the details of the server where the gateway is running. For example:

```
http://machine1234/PSIGW/PeopleSoftListeningConnector
```

4. In the Requesting Node field, enter `SOURCENODE`.
5. In the Message Name field, enter `EXAMPLE_REQUEST_MSG`.
6. From the Message 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. Open the PSFTmsg.txt file created by the subscription PeopleCode to view the body of the 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 PeopleSoft Application Designer, open the message *EXAMPLE_REQUEST_MSG* and add the following PeopleCode to the OnRequest message event:

```
/* get the body of the incoming message */
&xmlDoc = GetMessageXmlDoc();

/* and write it out to a file */
&theFile = GetFile("HttpRequest.txt", "w");
&theFile.writestring(&xmlDoc.GenXmlString());
&theFile.Close();

/* create the response message */
&msg = CreateMessage(Message.EXAMPLE_RESPONSE_MSG);

/* 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 */
&msg.SetXmlDoc(&xmlDoc);

/* and send the response message */
ReturnToServer(&msg);

```

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 message name and requesting node are present in the XML

```

<?xml version="1.0"?>
<IBRequest>
  <MessageName>EXAMPLE_REQUEST_MSG</MessageName>
  <From>
    <RequestingNode>SOURCENODE</RequestingNode>
  </From>
  <ContentSections>
    <ContentSection>
      <Data>
        <![CDATA[<?xml version="1.0"?><ConnectorTest>
          Testing the HTTPListeningConnector. Message body.
        </ConnectorTest>]]>
      </Data>
    </ContentSection>
  </ContentSections>
</IBRequest>

```

4. Click the Post button to sent the message to the integration gateway.
5. Check the Output section for the response. Compare the response with the PeopleCode used in the OnRequest event for the message. 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:

`MessageName:EXAMPLE_REQUEST_MSG`

`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 PeopleCode used in the OnRequest event for the message. 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?&MessageName=EXAMPLE_REQUEST_⇒
MSG&From=SOURCENODE`

Replace `<your_server_name>` with the details of the server where the integration gateway is running. For example:

`http://machine1234/PSIGW/HttpListeningConnector?&MessageName=EXAMPLE_REQUEST_⇒
MSG&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 sent the message to the integration gateway.
5. Check the Output section for the response. Compare the response with the PeopleCode used in the OnRequest event for the message. 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: #EXAMPLE_REQUEST_MSG#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-ENC="http://schema.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
    <ConnectorTest>
      <Text>
        Testing the HttpListeningConnector. SOAP Message.
      </Text>
    </ConnectorTest>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

5. Click the Post button to sent the message to the integration gateway.
6. Check the Output section for the response. Compare the response with the PeopleCode used in the OnRequest event for the message. 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.

```
/* create an XML document */
&xmldata = "<?xml version='1.0'?><ConnectorTest/>";
&xmlDoc = CreateXmlDoc(&xmldata);
```



```

/* add text to it */
&rootNode = &xmlDoc.documentelement;
&descNode = &rootNode.AddElement("TestNode");
&descNode.NodeValue = "This message will be sent to an HTTP server.";

/* send the document out as a sync message */
&xmlResponseDoc = SyncRequestXmlDoc(&xmlDoc, Message.EXAMPLE_REQUEST_MSG);

/* convert the response to an XML string */
&textString = &xmlResponseDoc.GenXmlString();

/* and echo it back to the user */
MessageBox(0, "", 0, 0, &textString);

```

2. In the PeopleSoft Pure Internet Architecture, open the node definition for TARGETNODE. Set the Connector ID to *HTTPTARGET*. Set the URL property value to the address of the HTTP server that will process the request.
3. Open the EXAMPLE_PAGE page, and click on the Test button. The HTTP response will be displayed in the resulting message box.

Sending SOAP Messages in HTTP Requests

To send a SOAP message in an HTTP request:

1. In PeopleSoft Application Designer, open the EXAMPLE_WORKREC record and add the following PeopleCode to the FieldChange event for the TEST field.

```

/* create a SOAP document */
&soapReq = CreateSOAPDoc();
&soapReq.AddMethod("TestNode", 1);
&soapReq.AddParm("Text", "This is a SOAP request.");

/* and send the document out in a sync message */
&request = &soapReq.XmlDoc;
&response = SyncRequestXmlDoc(&request, Message.EXAMPLE_REQUEST_MSG);

/* convert the response to a SOAP document */
&soapRes = CreateSOAPDoc();
&soapRes.XmlDoc = &response;

/* and echo the SOAP XML back to the user */
&strXml = &soapRes.GenXmlString();
MessageBox(0, "", 0, 0, &strXml);

```

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 8.4 node and a PeopleSoft 8.1 node.

Understanding the PeopleSoft 8.1 Connectors Examples

When sending a message from a PeopleSoft 8.4 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 8.4 System

To set up data on the PeopleSoft 8.4 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.
 - a. Open the properties and click 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`.
4. Create a subscription on the `EXAMPLE_PSFT_MSG` message called `ExampleSubscription`, with the following PeopleCode:

```
/* get the body of the incoming message */
&xmlDoc = GetMessageXmlDoc();

/* and write it out to a file */
&theFile = GetFile("PSFT81Message.txt", "w");
&theFile.writestring(&xmlDoc.GenXmlString());
```

```
&theFile.Close();
```

5. Open the *EXAMPLE_PAGE* page and add an *EditBox* to the page, setting the following properties:

Property	Value
Record name	<i>EXAMPLE_REC</i>
Field name	<i>EXAMPLE_CHAR</i>

6. Open the *EXAMPLE_WORKREC* record. Add the following *PeopleCode* to the *FieldChange* event for the *TEST* field:

```
/* get the buffer data */
&rowset = GetLevel0();

/* build a message */
&message = CreateMessage(Message.EXAMPLE_PSFT_MSG);

/* copy buffer data to the message */
&message.CopyRowset(&rowset);

/* send the message */
&message.Publish();

MessageBox(0, "", 0, 0, "Message sent.");
```

7. In the PeopleSoft Pure Internet Architecture, 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.
8. Add an inbound transaction to the new node with the following properties. After doing so, verify that the transaction is active and save it.

Field	Value
Transaction Type	Inbound Asynchronous
Request Message	<i>EXAMPLE_PSFT_MSG</i>
Request Message Version	<i>VERSION_1</i>

9. Add an outbound transaction to the new node with the following properties. After doing so, verify that the transaction is active and save it.

Field	Value
Transaction Type	Outbound Asynchronous
Request Message	<i>EXAMPLE_PSFT_MSG</i>
Request Message Version	<i>VERSION_1</i>

10. 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 8.4 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 8.4 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 8.4 node.
8. In the PeopleSoft Pure Internet Architecture, invoke the Gateway Administration servlet and add the PeopleSoft 8.1 node to the PeopleSoft handler.
9. Open the Message Monitor and verify that the *EXAMPLE_CHANNEL* is running.

Using the PeopleSoft 8.1 Target Connector

In the example presented in this section, you will use the PeopleSoft 8.1 target connector to send a message from a PeopleSoft 8.4 system to a PeopleSoft 8.1 system.

To send a message from a PeopleSoft 8.4 system to a PeopleSoft 8.1 system:

1. In the PeopleSoft Pure Internet Architecture on the PeopleSoft 8.4 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 8.4 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 8.4 system.

To send a message from a PeopleSoft 8.1 system to a PeopleSoft 8.4 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 8.4 system, open Integration Broker 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.

Example 4: Using the JMS Connectors

This section discusses using the JMS listening and JMS target connectors.

Understanding the JMS Connector Examples

The examples in this section are intended to be generic and independent of the JMS provider being used. Because of this, in certain steps general instructions are provided. The actual details of the task will depend on the provider being used – and may be rather involved. Please refer to the appropriate documentation.

The error queue is not configured in the examples. However, configuring the error queue may be desirable should issues arise while trying the examples.

The examples in this section focus on queues, but the process for using the JMS connectors when working with topics is essentially the same.

See Also

Chapter 6, “Using Listening Connectors and Target Connectors,” Working With the JMS Connectors, page 102

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 8.45 PeopleBook: Integration Testing Utilities and Tools, "Using the Send Master Utility," Using MQSeries 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 */
&result = PublishXmlDoc(&xmlDoc, Message.EXAMPLE_REQUEST_MSG);

MessageBox(0, "", 0, 0, "Message sent.");
```

4. In the PeopleSoft Pure Internet Architecture, open the node definition for *TARGETNODE*. Set the Connector ID to *JMSTARGET*. Set the values for the following properties:

Property	Value
JMSFactory	ExampleConnectionFactory.
JMSProvider	Name of the provider being used.
JMSUrl	Connection URL for the provider.
JMSQueue	ExampleQueue.
JMSUserName	The username on the JMS provider.
JMSPassword	The encrypted password for the user ID.

5. In the `integrationGateway.properties` file, set the `ig.jms.JMSTargetConnector.JMSProvider.Password` property to the same value as was used for the `JMSPassword` field in the previous step.
6. 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, open the message *EXAMPLE_REQUEST_MSG* and add the following PeopleCode to the OnRequest message event:

```

/* get the body of the incoming message */
&xmlDoc = GetMessageXmlDoc();

/* and write it to a file */
&theFile = GetFile("JMSRequest.txt", "w");
&theFile.writestring(&xmlDoc.GenXmlString());
&theFile.Close();

/* create the response message */
&msg = CreateMessage(Message.EXAMPLE_RESPONSE_MSG);

/* build the body of the response */
&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 */
&msg.SetXmlDoc(&xmlDoc);

/* send the response message */
ReturnToServer(&msg);

```

4. In the integrationGateway.properties file, uncomment the following line:

```
ig.jms.Queues=1
```

5. 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_REQUEST_MSG
ig.jms.Queue1.MessageVersion	VERSION_1
ig.jms.Queue1.RequestingNode	SOURCENODE
ig.jms.Queue1.DestinationNode	<the name of the local node >

6. Deploy and start the JMSListeningConnectorAdministrator servlet.

See [Chapter 6, “Using Listening Connectors and Target Connectors,” Using the JMS Listening Connector, page 104.](#)

7. Test the connector:

- a. Send a text message to the example JMS queue. Set the text of the message to:

```

<?xml version="1.0"?>
<ConnectorTest>
  <TestNode>Sending a message to the JMS Listening Connector.</TestNode>
</ConnectorTest>

```

- b. Check the message logs and the file named in the OnRequest PeopleCode. The message should be present in both.

Example 5: 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:

```
&xmldata = "<?xml version='1.0'?><ConnectorTest/>";

/* create an XmlDocument */
&xmlDoc = CreateXmlDoc(&xmldata);
&rootNode = &xmlDoc.documentelement;

/* add text to the document */
&descNode = &rootNode.AddElement("TestNode");
&descNode.NodeValue = "This message was written to a file.";

/* send the Xml in a sync message */
&xmlResponseDoc = SyncRequestXmlDoc(&xmlDoc, Message.EXAMPLE_REQUEST_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	<i>PUT</i>
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:

```
/* create an Xml document */
&xmldata = "<?xml version='1.0'?><ConnectorTest/>";
&xmlDoc = CreateXmlDoc(&xmldata);

/* make a sync request using the document */
&xmlResponseDoc = SyncRequestXmlDoc(&xmlDoc, Message.EXAMPLE_REQUEST_MSG);

/* get the response, and put it in a message box to be displayed to the user */
&textString = &xmlResponseDoc.GenXmlString();
MessageBox(0, "", 0, 0, &textString);
```

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 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: <i>sourcenodename.msgname.publication_id.xml</i>
METHOD	<i>GET</i>

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 6: Using the FTP Target Connector

This sections discusses how to use the FTP target connector to:

- Upload files to an FTP server.
- Download files from an FTP server.

Prerequisites

For the examples presented in this section, you must have an active FTP server, as well as an account on that server.

Uploading Files to FTP Servers

To upload a file to an FTP server:

1. In PeopleSoft Application Designer, open the EXAMPLE_WORKREC record and add the following PeopleCode to the FieldChange event for the TEST field:

```
/* create an XML document */
&xmldata = "<?xml version='1.0'?><ConnectorTest/>";
&xmlDoc = CreateXmlDoc(&xmldata);

/* add text to it */
&rootNode = &xmlDoc.documentelement;
&descNode = &rootNode.AddElement("TestNode");
&descNode.NodeValue = "This message will be uploaded to an FTP server.";

/* and send the document out as an async message */
&success = PublishXmlDoc(&xmlDoc, Message.EXAMPLE_REQUEST_MSG);

MessageBox(0, "", 0, 0, "Message sent.");
```

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	<i>PUT</i>
USERNAME	Enter the FTP server login ID.
PASSWORD	Enter the password for the login to the FTP server. This password must be encrypted. Use the Password Encryption Utility at the bottom of the page to encrypt the password, if necessary

3. In the PeopleSoft Pure Internet Architecture, open the EXAMPLE_PAGE page and click the Test button.

Login to the FTP server and check for the file. Open the file and verify the contents.

Downloading Files From FTP Servers

To download a file from an FTP server:

1. Create an XML file with the following contents and place the file on an FTP server.

```
<?xml version="1.0"?>
<ConnectorTest>
<TestNode>This message will be downloaded from an FTP server.</TestNode>
</ConnectorTest>
```

2. In PeopleSoft Application Designer, open the EXAMPLE_WORKREC record and add the following PeopleCode to the FieldChange event for the TEST field:

```
/* create an XML document */
&xmldata = "<?xml version='1.0'?><ConnectorTest/>";
&xmlDoc = CreateXmlDoc(&xmldata);

/* send the document out in a sync message */
&xmlResponseDoc = SyncRequestXmlDoc(&xmlDoc, Message.EXAMPLE_REQUEST_MSG);

/* convert the response message to an XML string */
&textString = &xmlResponseDoc.GenXmlString();

/* and echo it out to the user */
MessageBox(0, "", 0, 0, &textString);
```

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	<i>GET</i>
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 display in the message box.

Example 7: 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:

```
/* create an Xml document */
&xmldata = "<?xml version='1.0'?><ConnectorTest/>";
&xmlDoc = CreateXmlDoc(&xmldata);

/* add text to the Xml */
&rootNode = &xmlDoc.documentelement;
&descNode = &rootNode.AddElement("TestNode");
&descNode.NodeValue = "This message was sent via SMTP to an email account.";

/* send an async message containing the Xml */
&success = PublishXmlDoc(&xmlDoc, Message.EXAMPLE_REQUEST_MSG);
```

2. In the PeopleSoft Pure Internet Architecture, open the *TARGETNODE* node definition.
 - a. On the Node Definitions-Connectors tab, set the Connector ID to *SMTPTARGET*.
 - b. Set the following properties to the values indicated:

Property	Value
DestEmailAddress	Set this property to the email address to which the email will be sent.
SourceEmailAddress	Set this property to the email address from which you are sending the message.

3. Access the integrationGateway.properties file. Locate the following line and replace <mailServerName> with the name of the SMTP server.

```
ig.connector.smtptargetconnector.host=<mailServerName>
```

4. In the PeopleSoft Pure Internet Architecture, open the EXAMPLE_PAGE page and click the Test button to send the message.

Check the destination email account for the message. Since the message is being passed through one or more SMTP servers, there may be some propagation delay and the message might not be received immediately.

APPENDIX C

Developing Transformations

This appendix provides guidelines on how to create and use Application Engine Transform programs and the PeopleSoft Integration Broker transformation engine to transform an XML data structure into almost any desired format. This appendix discusses:

- The essential elements for creating transformations.
- A transformation example for integration between two PeopleSoft nodes.

Working with the Essential Elements of Transformations

This section discusses how to:

- Extract a PeopleSoft message.
- Create application engine programs for transformation processing.
- Associate transformations with transactions.
- Perform data translation using codesets, codeset groups and codeset values.
- Implement codesets using XSLT or PeopleCode.
- Associate PeopleSoft nodes with codeset groups.
- Use XML and XSLT to create transformations.
- Debug transformation programs.
- Use non-XML data in transformation programs.
- Use the TransformData object.
- Filter transformations.
- Create data used during an XSLT step in an application engine program.
- Bypass PeopleSoft Integration Broker to invoke transformations.

Extracting PeopleSoft Messages

Being able to transform data into new formats gives you an extremely flexible way of connecting a PeopleSoft application to an third-party system or connecting two different PeopleSoft applications together. These connections between the differing systems are known as integration points. Creating these integration points is first accomplished by determining what type of data the sending system produces, as well as knowing what type of data the receiving system is expecting.

To better process all messages in a generic way, PeopleSoft Integration Broker uses the Extensible Markup Language (XML) when passing data throughout the system. Since PeopleSoft Integration Broker is expecting XML, you need to know how the message will look when represented as XML.

This section describes how to extract XML for an asynchronous message from a source (publishing) node and If an Integration Point is needed to connect two PeopleSoft Application's, this process would need to be repeated to retrieve the destination (receiving) node's XML message representation as well.

Retrieving XML Representations of Messages

To retrieve an XML representation of a message, you can send a test message to the integration gateway and use the Integration Broker Monitor to view the XML representation of the message sent to the gateway.

Before you generate a test message you must set up an appropriate transaction for the message.

To create a test message:

1. Start PeopleSoft Application Designer and open the message that you want to transform
2. Right-click the message version to use and select Create Test Message...

The Create Test Message tab displays for the message version you selected. The display shows the structure of the message and the fields that will be included.

3. To enter a value for a field, right-click or double-click the a field and enter a value.

Note. For a record to appear in the test message, at least one field must contain a value, otherwise the record will not be generated.

4. To exclude a field, right-click the field and select Delete.
5. Click the Apply button.

To view and extract the XML representation of the message:

1. Login to the PeopleSoft Pure Internet Architecture.
2. Select PeopleTools, Integration Broker, Monitor, Monitor Message.

A summary of the channels that have received asynchronous messages displays by default. Look in the Done column of the channel where the message was published and click on the number shown. The page updates to show all messages with a status of *Done*.

3. Click the Details link for the desired message. A new browser window opens and displays the properties for the selected message.
4. Click the XML Message Viewer tab. The XML representation of the message displays in the text box.
5. Select the contents of the text box and save it to a file titled <message name>.xml.

Creating Application Engine Programs for Transformation Processing

The transformation engine contained within the PeopleSoft Integration Broker utilizes PeopleSoft Application Engine to handle a majority of the processing. This means that the creation of a transform program takes place through PeopleSoft Application Engine Designer.

During runtime the transformation engine handles each step and action that you set up in PeopleSoft Application Engine in a sequential manner. As a result, the first step in the transform receives the original message and each subsequent step receive the output of the preceding step. The ability to create any number of PeopleCode or XSLT steps provides extreme flexibility when implementing transforms.

When you add a new action to a step, the action defaults to SQL and you must change the action type to PeopleCode or XSLT. You use the PeopleCode or XSLT actions to filter messages, perform structural transforms, execute codeset functionality and generate virtually any desired format.

To create an application engine program:

1. Start PeopleSoft Application Designer.
2. From the File menu, click New.
3. In the New Definition box, click App Engine Program and click OK.
4. From the File menu, click Definition Properties. The Program Properties dialog box displays.
5. Click the Advanced tab and from the Program Type drop-down list, select Transform Only.
6. Right-click on the Step, and choose Insert Action.
7. From the drop-down list, select PeopleCode or XSLT, based on the language that you're using to develop the transformation program.

To create a PeopleCode Actions, double-click the action. The appropriate editor displays, based on the language you selected. If XSLT is the desired Action, PeopleSoft recommends that you construct the XSLT in an outside editor. The use of any simple text editor should suffice, or you can use a third-party XSLT designer. After you create the XSLT in your editor, paste the contents into the XSLT Action.

Associating Transformations with Transactions

Before you can associate a transformation with a transaction, a relationship must exist.

Depending on how a relationship is setup, you can set a transform on either the Request or Response. Choosing when to perform a transformation is completely up to the developer. Usually, most transactions will require a transform on both Request and Response, because neither system speaks each others message format.

To associate a transformation with a transaction:

1. Login to the PeopleSoft Pure Internet Architecture.
2. Select PeopleTools, Integration Broker, Relationships.
3. Select the relationship that needs a transformation.
4. Click the Trans Modifiers tab.
5. Click the Edit link for the transaction that requires a transform.
6. In the Transformation section, click the Lookup button to choose a transformation required to complete the transaction.

Performing Data Translation Using Codeset Repositories

This section discusses creating and implementing code sets to perform data translation.

Understanding Data Translation

Being able to transform a message's structure is extremely important. But what about being able to transform the data contained within that message? For instance, you could have system A that represents all of its purchase order status codes in an easily readable format, using words such as Open and Complete. On the other hand, you could want to send a message to system B which represents all of its purchase order status codes as numerical values. If either of these systems were to pass along their own internal status codes within messages, the systems would have no idea how to interpret them.

PeopleSoft Integration Broker can apply a transform program to translate these values so each participating system can understand them. The translating of this data is achieved through the use of codeset repositories, which include codesets, codeset groups and codeset values.

Note. Data translation is best-suited for modifying message content, rather than structure.

By leveraging the this functionality, you can solve:

- Direct one-to-one mappings.
- One-to-many mappings.
- Many-to-one mappings.
- Many-to-many mappings.

Codeset repositories not only allow the ability to create simple one-to-one data value mappings, they also give you the opportunity to create much more complicated mappings. For instance, since system B represents all status codes as numerical values, it will need to convert all status codes to a readable format when communicating with system A. What if system A has offices all across the globe, with each of its software installations utilizing the local language? Would passing a status code of Open to one of the offices that was expecting Abierto really be of any benefit? To resolve this, system B would not only have to use the status code, but a possible language code contained within the message. By using both of these values together, the appropriate system A status code can be generated.

Understanding Codeset Repositories

The following elements compose a codeset repository:

Element	Description
Codeset group	Contains a list of significant data fields and their values that a node might send in a message.
Codeset	Contains a specific set of match/name value pairs from an existing codeset group that are used to translate values between multiple systems.
Codeset values	Contains the named value you predefine as the return value.

Once you create codesets and codeset values, you can invoke them from your translation program.

You must initially develop these elements in the following order:

1. Codeset group (for each system participating in the translation).
2. Codeset for the system on which the translation will occur.
3. Codeset values for the codeset.

Creating Codeset Groups

You must create codeset groups each system participating in the translation. However, you need to specify name/value pairs only for the system on which the translation is to occur. So, if you are performing integrations between PeopleSoft SCM and PeopleSoft CRM systems and you want to map data values from the PeopleSoft CRM system into the PeopleSoft SCM system, you would:

- Define codeset groups for both the PeopleSoft SCM and PeopleSoft CRM systems.
- Define name/value pairs only for the codeset you created for the PeopleSoft SCM system.

Codesets and their output values are associated through codeset groups. When you create a codeset group, you must enter match name and match values for data. For example, your system could feature a Status setting with three possible values of 0, 1000 and 2000.

Code Set Group Name: PSFT_SCM

Customize Find View All  First 1-6 of 6 Last		
	Match Name	Match Value
1	LANG	en_us
2	LANG	fr
3	LANG	sp
4	STATUS	0
5	STATUS	1000
6	STATUS	2000

Codeset group for a PeopleSoft SCM system

Code Set Group Name: PSFT_CRM

Customize Find View All  First 1 of 1 Last		
	Match Name	Match Value
1		

Codeset group for a PeopleSoft CRM system

To create a codeset group:

1. Select PeopleTools, Integration Broker, Codeset Groups.
2. Click the Add a New Value tab, and in the Code Set Group Name field enter a name for the group.
Make the name as descriptive as possible, because it will be used in variety of other places. For example, If a few PeopleSoft SCM nodes are to share this codeset group, call the group PSFT_SCM.
3. Click the Add button. A Code Set Group Name page displays.
4. For the codeset group for the system performing translations, enter Match Name and Match Values as appropriate.
5. Click the Save button when finished.

Creating Codesets

When you create a codeset, you're creating a subset match names and match values from a particular codeset group. The graphic in the previous section show that the codeset group PSFT_SCM contains two data fields, or match names, called *LANG* and *STATUS*. Each of these match names has three possible values, or match values: the *LANG* match name has the possible match values of *en_us*, *fr* and *sp*; the *STATUS* match name has the possible match values of *0*, *1000* and *2000*.

When you create a codeset you create a record that specifies specific match names (data fields) and values from a particular codeset group to use in translation.

For example, you could create a codeset based on the PSFT_SCM codeset group that contains only the *STATUS* match name and only two of the three match values defined in the codeset group. The following graphic shows such a codeset.

The screenshot shows a web-based interface for configuring a codeset. At the top, the 'Code Set Group Name' is 'PSFT_SCM' and the 'Code Set Name' is 'STATUS_CHANGE'. Below this is a table with two columns: 'Match Name' and 'Match Value'. The table has two rows. The first row shows 'STATUS' as the match name and '0' as the match value. The second row shows 'STATUS' as the match name and '1000' as the match value. To the right of each match value is a search icon and a plus/minus button. The interface also includes a 'Customize' button and a 'Find' button.

	Match Name	Match Value
1	STATUS	0
2	STATUS	1000

A codeset based on the PSFT_SCM codeset group name

To create a codeset:

1. Select PeopleTools, Integration Broker, Codesets.
2. Click the Add a New Value tab.
3. In the Code Set Group Name field, enter the name of the codeset group on which the codeset is based.
4. In the Code Set Name field, enter a descriptive name for the codeset.
5. Click the Add button.
6. In the Match Name field, select a match name from the codeset group.
7. In the Match Value field, select a match value from the codeset group.
8. Click the plus button (+) and repeat Steps 6 and 7 to enter all possible match name /value pairs that can pass through the codeset.
9. Click the Save button.

Creating Codeset Values

A codeset value is a return value that you define. When you run a translation program, the program outputs the return value as a result of matching a combination of codeset match values.

For instance, you could define a codeset value so that if a translation program encountered a status code of *Open* on the PeopleSoft CRM system, it would translate and return a value of *0* on the PeopleSoft SCM system.

The first step to creating codeset values is to add a codeset value to the database. In this step you specify the source codeset group, a codeset from that source codeset group, and a target codeset group.

Codeset Values

Find an Existing Value
Add a New Value

From Group:

Code Set Name:

To Group:

Adding a codeset value to the database

The next step is to enter a description for the record, which typically describes what the combination will resolve.

Next, you choose the desired combination of Match Name/Value pairs, enter in a descriptive Return Name and specify the value to output.

From Group: PSFT_SCM

Code Set Name: STATUS_CHANGE

To Group: PSFT_CRM

Code Sets
Find | View All
First ◀ 1 of 1 ▶ Last

Description: + -

Select	Match Name	Match Value
☒	LANG	en_us
☐	LANG	fr
☐	LANG	sp
☒	STATUS	0
☐	STATUS	1000
☐	STATUS	2000

Code Set Values
Customize | Find | View All
First ◀ 1 of 1 ▶ Last

Return Name	Return Value
RET_STATUS	Open + -

Creating codeset values

The previous graphic shows that the description name used for this combination is *Status Open (English)* and that the Match Name/Value pairs of *LANG/en_us* and *STATUS/0* are selected. The next step is to choose a Return Name and a Return Value. This example shows using multiple input (match) values to determine one return value. If multiple Return Name/Value pairs are specified for a given combination of Match Name/Value pairs, each of those Return Names must be unique.

In this example, both *LANG* and *STATUS* values pass through this codeset, so rules are being set to handle each possible combination. These rules are used to generate the desired output. In the example shown, if a language code of *en_us* and a status code of *0* are found, the value of *Open* should be returned.

These steps need to be repeated to handle each combination the codeset will encounter (for example, *LANG=en_us* and *STATUS=1*, *LANG=en_us* and *STATUS=2*). To create a new combination, click the plus (+) button next to the Description field. Make sure to use the same return name values for each combination, because these names will be referenced in both the PeopleCode and XSLT Steps in the application engine program.

You may set a default value in the event that an unexpected combination comes through. First, make sure that each match name in the codeset has a corresponding blank Match Value associated with it. In the previous example both *LANG* and *STATUS* would need to have a blank entry as the match value.

From Group: PSFT_SCM
Code Set Name: STATUS_CHANGE
To Group: PSFT_CRM

Code Sets Find | View All First 2 of 2 Last

Description: Default + -

Select	Match Name	Match Value
☒	LANG	
☐	LANG	en_us
☐	LANG	fr
☐	LANG	sp
☒	STATUS	
☐	STATUS	0
☐	STATUS	1000
☐	STATUS	2000

Code Set Values Customize | Find | View All First 1 of 1 Last

*Return Name	Return Value
RET_STATUS	Pending

Blank match values defined for LANG and STATUS to set a default.

If an unexpected combination of match name/value pairs is encountered during processing and a default value is not set, the transform will error out. You need to determine if foreign data comes along with the message, whether you want to pass it along, and act accordingly. For example, if PeopleSoft SCM has a variety of status codes that PeopleSoft CRM does not care about, perhaps status codes that represent awaiting manager approval or items on back order, these could easily be rolled up into a single default value of *Pending* for PeopleSoft CRM.

To create codeset values:

1. Select PeopleTools, Integration Broker, Codeset Values.
2. Click the Add a New Value tab.
3. In the From Group field, enter the codeset group name for the source node.
4. In the Code Set Name field, enter the name of the codeset to use.
5. In the To Group field, enter the codeset group name for the target node.

6. Click the Add button.
7. Check the select box next to one or more match/value pairs to include in the codeset value.
8. In the Return Name field, enter a description name for the return value.
9. In the Return Value field, enter a value to be returned by the translation program when it encounters the combination of match/value pairs selected.
10. Click the Save button.

Implementing Codesets Using XSLT or PeopleCode

Once setup is complete you implement the transformation through XSLT or PeopleCode. Each has approach has a different implementation style that enables you to choose how to solve the needs of each unique transform.

Implementing Codesets Using XSLT

When codesets are executed through XSLT, they are handled in a second pass of the XML document. The first pass performs a structural transform of the data and then continues on to execute the codeset functions found within the XML document. This means that during the structural transform of an XML document, the XSLT must output PeopleSoft codeset functions.

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

  <xsl:template match="Header">
    <NewHeader>
      <xsl:apply-templates/>
    </NewHeader>
  </xsl:template>

  <xsl:template match="STATUS_CODE">
    <STATUS>
      <psft_function name="codeset" codesetname="STATUS_CHANGE">
        <parm name="LANG"><xsl:value-of select="../LANGUAGE_CODE"/>
        </parm>
        <parm name="STATUS"><xsl:value-of select="."/></parm>
        <value name="RET_STATUS" select="."/>
      </psft_function>
    </STATUS>
  </xsl:template>

</xsl:stylesheet>
```

This example states the following: “When a node with a name of ‘STATUS_CODE’ is encountered, output the data contained within the ‘STATUS_CODE’ template.”

Here is sample input:

```
<?xml version="1.0"?>
<Header>
  <LANGUAGE_CODE>en_us</LANGUAGE_CODE>
  <STATUS_CODE>0</STATUS_CODE>
```

```
</Header>
```

When this input is run through the above XSLT, the first pass yields the following:

```
<?xml version="1.0"?>
<NewHeader>
  <STATUS>
    <psft_function name="codeset" codesetname="STATUS_CHANGE">
      <parm name="LANG">en_us</parm>
      <parm name="STATUS">0</parm>
      <value name="RET_STATUS" select="."/>
    </psft_function>
  </STATUS>
</NewHeader>
```

The second codeset processing pass with ultimately yield the following:

```
<?xml version="1.0"?>
<NewHeader>
  <STATUS>Open</STATUS>
</NewHeader>
```

Since codeset processing occurs on the second pass of a XSLT transform Step, there must be a way for the transformation engine to know exactly where to perform the data translations. The transformation engine recursively finds each occurrence of the **psft_function** nodes and performs the lookups. Which means that the psft_function nodes can be nested within each other, giving you the ability to use the output from one codeset lookup as the input of another.

The psft_function node currently has four attributes to utilize.

Attribute	Description
name	The only name allowed here currently is codeset. This name tells the transformation engine to start processing a codeset lookup.
codesetname	This attribute tells the transformation engine which codeset to use while performing the lookup. If no name is specified an error will be thrown.
dest	(Optional). Use this attribute to override the transaction's destination PeopleSoft Node. If this is used, make sure the specified node exists and it has an association to a codeset group.
source	(Optional) Use this attribute to override the transaction's source PeopleSoft node. If this is used, make sure the specified Node exists and it has an association to a codeset group.

Contained within each `psft_function` node must be at least one parameter node. A parameter node acts as the vehicle for specifying the Match Name/Value pairs, which means there could be many of them contained within a `psft_function` node. There is only one attribute associated with this node.

Attribute	Description
Name	Contain the match name for a codeset. In the example earlier in this document the names used were LANG and STATUS.

Just as the parameter node, there must be at least one value node contained within the `psft_function` node. Without value nodes, the transformation engine would have no idea where to insert the returned data from the codeset lookup. If there is more than one return value expected for a codeset, make sure to create the necessary value nodes as needed. There are four attributes that can be utilized when using the value node:

Attribute	Description
name	This attribute contains the return name to use when processing the value node. In the example earlier in this document, the return name specified for the codeset was <i>RET_STATUS</i>.
select	This attribute contains the XPath to use when outputting the data. In most cases the XPath will be “.”, which means current working node. When the transformation engine encounters “.”, it replaces the <code>psft_function</code> node with the returned data. Much more complex XPath expressions can be used to determine where to generate the output. The <code>psft_function</code> node will be removed after the second pass.

Attribute	Description
createIfDNE	This attribute can have two values, either <i>yes</i> or <i>no</i>. The default value is <i>no</i>. If a value of <i>yes</i> is found, the transformation engine creates the nodes specified in the XPath. For example, if an XPath of: <pre>../Header/Address/Name</pre> is encountered, the transformation engine creates any part of that path that is not found and inserts the return value there.
createNodeFromValue	This attribute can have two values, either <i>yes</i> or <i>no</i>. The default value is <i>no</i>. If <i>yes</i> is found, the transformation engine creates a new node with the return value used as the name. For example, with an XPath of: <pre>../Header/Shipping</pre> and a return value of <i>FedEx</i>, the transformation engine will navigate through the XML document using the XPath and create a node with the name of <i>FedEx</i>. Any child nodes found within a value node with this flag turned on will be appended to the newly created node.

Implementing Codesets Using PeopleCode

Using the same input file, the above example can be used to generate the same output via PeopleCode. The function used to accomplish this is **FindCodeSetValues**.

```
2Darray = FindCodeSetValues(    String    CodeSetName,
                               2Darray    nameValuePairsToLookUp,
                               String      sourceNodeName,
                               String      destinationNodeName);
```

The FindCodeSetValues function takes four parameters and returns a two-dimensional array. The 2DarrayIs contains a 1:N occurrences of the return name/value pairs.

Parameter	Description
CodeSetName	Tells the transformation engine which codeset to use while performing the lookup.
NameValuePairsToLookUp	An array that contains 1:N occurrences of match name/value pairs.

Parameter	Description
sourceNodeName	Contains the source PeopleSoft node name needed for the codeset lookup. In most cases, the value of &incomingData.SourceNode is passed.
destinationNodeName	Contains the destination PeopleSoft node name needed for the codeset lookup. In most cases, the value of &incomingData.DestNode is passed.

The following example demonstrates implementing codesets using PeopleCode.

```

/* 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*/
Local string &langCode = &tempDoc.DocumentElement.FindNode
    ("LANGUAGE_CODE").NodeValue;
Local string &statusCode = &tempDoc.DocumentElement.FindNode
    ("STATUS_CODE").NodeValue;

/* 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 &statusCodeNode = &tempDoc.DocumentElement.AddElement
            ("STATUS");

        /* Since this is a 2D array, get the Return Value*/
        &statusCodeNode.NodeValue = &outAry [1][2];
    End-If;
End-If;

```

Associating PeopleSoft Nodes with Codeset Groups

During runtime, the transformation engine is passed both the source and destination PeopleSoft Nodes names. From this information, the engine performs a lookup to determine the codeset group with which each PeopleSoft node is associated. Without this lookup the codeset data would be irretrievable, because codeset values are associated through codeset groups, not with individual PeopleSoft nodes.

Each PeopleSoft node that must perform transformations must be associated with a codeset group. Since the default local node handles most of the publishing and subscribing, it is extremely important to make sure that it is associated with a codeset group. For example, if the default local node is a PeopleSoft SCM node, make sure it's associated with a PeopleSoft SCM codeset group.

To associate a PeopleSoft node with a codeset group:

1. Select PeopleTools, Integration Broker, Node Definitions.
2. Select a PeopleSoft node.
3. Click the Node Info tab, if not already selected.
4. In the Code Set Group Name field, enter a code set group name.
5. Click the Save button.

Understanding Technologies Used for Transformations

This section highlights technologies used to create and work with transformations.

Extensible Markup Language (XML)

A more detailed explanation of this technology is found at <http://www.w3.org/TR/REC-xml>.

Being able to represent data using XML bring many advantages:

- Message data must follow the strict well-formed structure of the XML standard.
- XML representation of the data is extremely readable.
- Data is easily transported to external systems.

Each of these advantages allows almost every system the ability to complete business transactions via the web. The problem with this whole concept is that when systems agree to communicate with one another it is usually after the fact. These systems have already completed their applications and simply need some intermediary to convert the data they are currently generating into any systems desired format. This is where the Integration Broker comes into play and why learning XML becomes so important.

The following terminology is used when discussing and working with XML:

Attribute

An XML attribute is a placeholder for data. This is simply another way of representing data with an XML structure.

One way to represent data is:

```
<Name>
  <First>Jane</First>
  ...<Last>Doe</Last>
</Name>
```

Instead of creating two separate nodes to store data, they can be rolled into one node declaration:

```
<Name first="Jane" last="Doe"/>
```

An attribute is found within the node declaration, with its value encapsulated in double quotes. This gives the developer an option of decreasing the overall size of the XML Document.

Document

A Document is the complete representation of a data structure using XML notation, i.e. a PeopleSoft message. An XML Document contains an XML declaration followed by a root node, which can contain a varying number of child nodes.

This XML declaration must be the following for all PeopleSoft messages:

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

Namespace

This allows you to associate a set of node and or attribute names with a unique identifier.

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

In this example, whenever a node or attribute name starts with *xsl* it must be part of the collection located at the specified Uniform Resource Identifier (URI). What this does, is set a few rules on how certain node and attribute names will be used within a Document. If you decide to associate a node or attribute with a given URI, and it is not part of that collection, a parse exception will be thrown.

Node

An XML node, also referred to as a *tag*, is a placeholder for data and child nodes. There are a few important rules that need to be followed when constructing XML nodes.

- A node must always start with an “<” character and end in a “>” character.
- Only integers, letters both uppercase and lowercase, and the “_” character are allowed in a node name. All other characters are not allowed and the parser will throw an exception when the XML is parsed.
- Nodes must never start with an integer.
- For every node that is created, there must be a corresponding end node associated with it. For example:

```
<Quantity>15</Quantity>
```

- The corresponding end node must start off with the “</” characters.
- The following characters are reserved characters for the XML standard: &, <, >, ‘ and “. If these characters are to be used as data within a node, they must first be escaped as &, <, >, ', ", respectively.

```
<Name>Jason &amp; Son&apos;s</Name>
```

The output will look like the following:

```
<Name>Jason & Son's</Name>
```

Parser

This is the software that is used to read in and validate an XML data structure. Currently, PeopleSoft Integration Broker uses the parsers created by the Sun Apache Group located at <http://xml.apache.org>.

Extensible Stylesheet Language Transformations (XSLT)

A more detailed explanation of this technology is found at <http://www.w3.org/TR/xslt>.

When you work with XSLT keep the following points in mind:

- XSLT is driven by XML. This means that XML is the only valid data structure that XSLT will accept as input.
- XSLT is a recursive language that breaks up the processing of nodes into templates. These templates allow a developer to set a number of actions to occur when a specified node or attribute is encountered as the input XML is processed. Breaking up processing into these templates not only improves performance, it also makes an XSLT much easier to read and update.
- XSLT utilizes both XPath and Boolean logic when processing XML. XPath is the language used to represent the path to a node or attribute from a specific location within an XML Document. Think of it this way, when trying to navigate to a specific directory using DOS, one must type in the specific directory to which they wish to move relative to their current position. XPath takes a similar approach. More detail can be found at <http://www.w3.org/TR/xpath>.
- Every XSLT is an XML Document and as such must follow the World Wide Web Consortium (W3C) XML 1.0 standard. Basically, every node that is declared must have a corresponding closing node.

Each XSLT used within PeopleSoft Integration Broker must first start out with the following two lines:

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

These two lines contain the XML declaration as well as the XSLT namespace to use while processing.

Note. If you use a third-party XSLT tool during development, please beware; they generally generate a different namespace URI. Make sure to always point the “xsl” namespace to <http://www.w3.org/1999/XSL/Transform>. Unexpected output will be produced if this namespace is not used.

Debugging Transformations

For debugging, log files can be generated to view the output of transformation steps.

Enabling Logging

To enable logging:

1. Open the psappsrv.cfg file found on the application server. The location is usually as follows for Windows:

```
<drive>:\<PS_HOME>\APPSERV\<domain name>
```

2. Search for the TraceAE flag. It should be on line 279.
Bring down the application server and reboot it.
3. Set the flag to 8192.

4. Reboot the application server.

Once this flag is turned on, trace files will be created in the <drive>:\<ptversion>\APPSERV\<domain name>\LOGS directory with a file extension of AET. Open these files in any text editing software (NotePad) and one can see the original XML structure as it entered the transformation engine, as well as the output of the XML as it passed through each step.

Sample .AET Trace File

The following code shows a sample .AET trace file:

```
PeopleTools 8.40-IB3 -- Application Engine
Copyright (c) 1988-2001 PeopleSoft, Inc.
-- All Rights Reserved
-- Database: QE840IB3 (SqlServer)

-- 11.28.55 (PC_CS_EXMPL.MAIN) (XSLT)
-- Original XML
<?xml version="1.0"?>
<Header><LANGUAGE_CODE>en_us</LANGUAGE_CODE><STATUS_CODE>0</STATUS_CODE>
</Header>
-- 11.28.56 (PC_CS_EXMPL.MAIN.Step01) (PeopleCode)
-- XML
<?xml version="1.0"?>
<NewHeader><STATUS>Open</STATUS></NewHeader>
```

Using Non-XML Data

The transformation engine requires all data to be represented as XML. If non-XML data is expected to pass from step to step during a transformation, it will need to be wrapped in XML. This keeps the transformation engine focused on one data structure type, alleviating it from having to handle a multitude of data structures.

```
Jane Doe, 123 Happy Street, Pleasantville, CA, 95266
John Doe, 3894 West Eastland, Center Outskirts City, CA, 93944
```

If the preceding data needed to be passed through the system, it would need to look as follows:

```
<?xml version="1.0"?>
<data psnonxml="yes">
  <![CDATA[Jane Doe, 123 Happy Street, Pleasantville, CA, 95266
John Doe, 3894 West Eastland, Center Outskirts City, CA, 93944]]>
</data>
```

The root node is set as data and the required attribute psnonxml with a value of “yes” must be present. The data will then be placed inside the CDATA node. The purpose of a CDATA node is simple, it tells the parser to leave its contents alone. A word of caution though, certain data contained within a CDATA node can cause the parser to fail during load. If a developer tries to store another XML document within a CDATA node that contains an XML Declaration, this will fail. Only one XML Declaration can exist in an XML Document.

Using the TransformData Object

The TransformData object provides access to important data during a transformation. The TransformData object has the following properties:

Property	Description
XMLDoc	Read/Write. Contains the PeopleCode object XmlDoc, which contains the contents of a message. The message structure can be manipulated via the XmlDoc object.
Status	Read/Write. Default value is <i>0</i> , which indicates success. This flag allows you to change how a transformation should proceed once it leaves a Step. Setting this flag to <i>1</i> during a PeopleCode Step and changing the root node to <i>Filter</i> will tell the transformation engine that a Step failed a filter. Setting this flag to <i>2</i> and changing the root node of the XmlDoc to <i>Error</i> will tell the engine that an error occurred. In either case, the message will error out and not continue processing.
SourceNode	Read only. Contains the name of the source PeopleSoft Node.
DestNode	Read only. Contains the name of the destination PeopleSoft Node.
SourceMsgVersion	Read only. Contains the source message version.
DestMsgVersion	Read only. Contains the destination message version.
SourceMsgName	Read only. Contains the source message name.
DestMsgName	Read only. Contains the destination message name.

To access the object you must pull the global TransformData into a local variable.

```
Local TransformData &incomingData = %TransformData;
```

Once this is completed, you can reference the above properties and manipulate the XmlDoc object. The XmlDoc is extremely important because it contains the message data as it passes through each transformation Step.

Filtering Messages

You can filter messages that enter PeopleSoft Integration Broker using the transformation engine. If a message currently utilizes a transform you need to add a PeopleCode Step to the application engine program. This step is usually placed as the first step in a transformation, but you can decide at which step filtering should occur.

PeopleCode Example

The following code shows an example of PeopleCode you would place on a Step to filter messages:

```
/* 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 status code */
Local string &statusCode = &tempDoc.DocumentElement.FindNode
("STATUS").NodeValue;
```



```

/* Make sure the status code is "Open" */
If &statusCode <> "Open" Then

    /* Clear out the doc and put in the "Filter" root node */
    If (&tempDoc.ParseXmlString("<?xml version=\"1.0\"?>
        <Filter/>")) Then

        /* Set the root node value to be the reason for failing filter */
        &tempDoc.DocumentElement.NodeValue = "Status Code of '"
            | &statusCode | "' is not allowed.";

        /* Set the status of the transformation to 1 for failed filter*/
        &incomingData.Status = 1;

    End-If;
End-If;

```

To properly filter out a message, the transformation engine must see two things when leaving a step:

1. The Status flag on the TransformData object must be set to 1.
2. The XmlDocument object must contain one node with its value containing the reason for failing the filter. For standards reasons, PeopleSoft recommends that you call the root node *Filter*.

Raising Errors During PeopleCode Steps

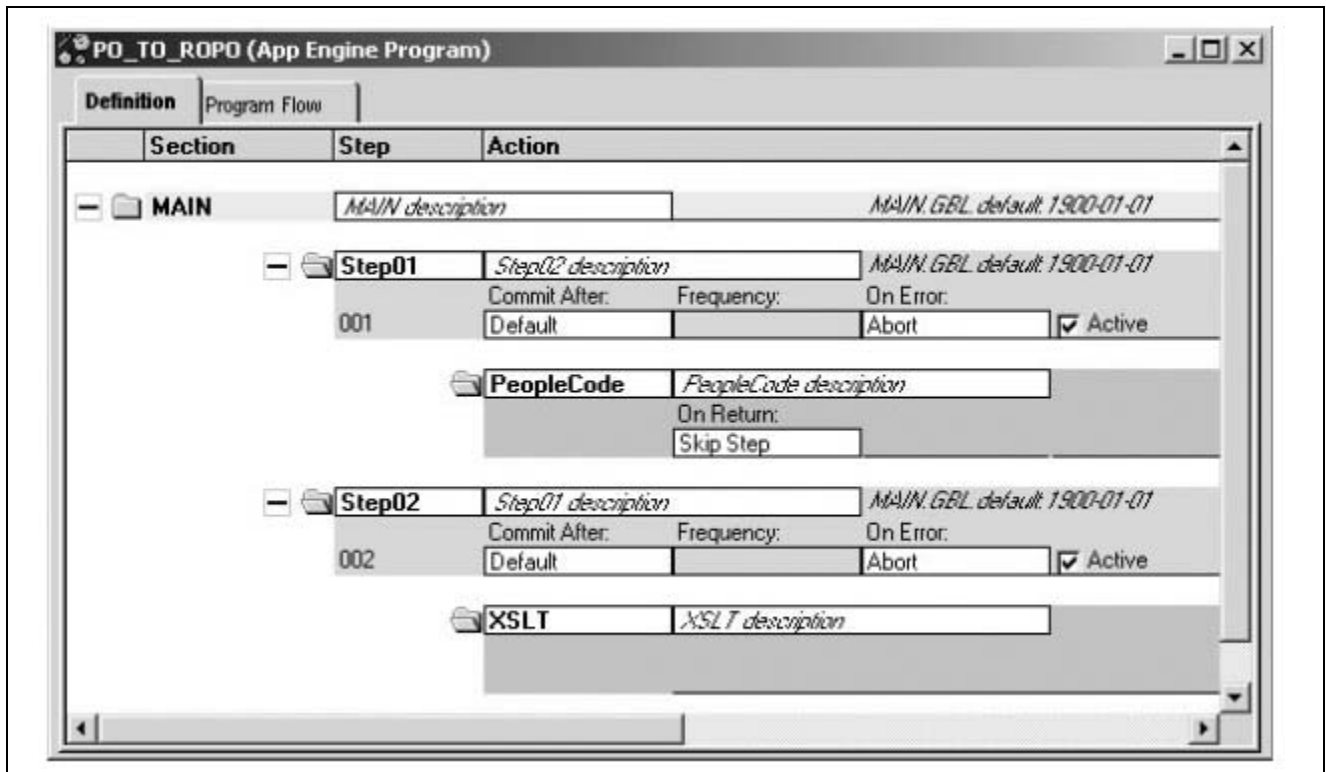
You might also decide to simply error out from a transformation. During PeopleCode processing, if something were to programmatically fail, it would be nice to catch an error and return a readable message. Currently, the transformation engine catches any PeopleCode errors encountered and report them appropriately. However, would someone reading the message monitor want to see something readable like “Error creating status code output Document” or the actual error of “XML parser error ParseXmlString Fatal Error: at file Integration Server line: 1 column: 30 message (159,5)”? This is completely up to your discretion and can be accomplished in a similar fashion to filtering.

1. The Status flag on the TransformData object must be set to 2.
2. The XmlDocument object must contain one node with its value containing the reason for the error. For standards reasons, it is recommended to call this root node *Error*.

Creating Data to Be Used During XSLT Steps

The transformation engine does not support callouts during the processing of XSLT steps. If you need to access stored data or need to format data in a way that falls out of scope from the XSLT functionality, you must create it through a PeopleCode Step. This PeopleCode Step can be placed either as the first step in a transform or in a Step prior to the Step which requires the data.

Note. If you decide to place this PeopleCode Step as the first one, you must make sure that each proceeding Step properly copies the data. Remember, XSLT only copies data to the output if it has been explicitly defined, if a template does not exist to copy that data, it will not pass to any following Steps.



A PeopleCode Step followed by an XSLT Step

The PeopleCode Step found in this example is used to access the current system date. PeopleSoft recommends that you use the name of *psft_workingstorage* as the node name that will contain the appropriate data. Any valid XML structure can be stored in this location. Take a look at the following example:

Input XML:

```
<?xml version="1.0"?>
<Header>
  <LANGUAGE_CODE>en_us</LANGUAGE_CODE>
  <STATUS_CODE>1000</STATUS_CODE>
</Header>
```

PeopleCode step:

```
/* 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);
```

Output structure after processing PeopleCode step:

```
<?xml version="1.0"?>
<Header>
  <LANGUAGE_CODE>en_us</LANGUAGE_CODE>
  <STATUS_CODE>0</STATUS_CODE>
  <psft_workingstorage>
    <sysdate>2002-01-02</sysdate>
  </psft_workingstorage>
</Header>
```

XSLT step to access psft_workingstorage data:

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

  <xsl:template match="Header">
    <Root>
      <xsl:attribute name="date"><xsl:value-of
        select="psft_workingstorage/sysdate"/>
      </xsl:attribute>
      <xsl:apply-templates/>
    </Root>
  </xsl:template>

  <!-- This will copy a node/attribute and it's children -->
  <xsl:template match="node() | @">
    <xsl:copy>
      <xsl:apply-templates select="node() | @"/>
    </xsl:copy>
  </xsl:template>

  <!-- Ignore these fields -->
  <xsl:template match="processing-instruction() | psft_workingstorage">
    <!-- do nothing -->
  </xsl:template>

</xsl:stylesheet>
```

Simple XSLT which ignores the psft_workingstorage node during the copy, but uses the system date to create a required attribute.

Final output:

```
<?xml version="1.0"?>
<Root date="2002-01-02">
  <LANGUAGE_CODE>en_us</LANGUAGE_CODE>
  <STATUS_CODE>1000</STATUS_CODE>
</Root>
```

Bypassing PeopleSoft Integration Broker to Invoke Transformations

If you need the ability to invoke a transformation without going through PeopleSoft Integration Broker, use the Transform PeopleCode function. This function is defined as:

```
XmlDoc = Transform(    String    InputXMLString,
                      String    AETransformName,
                      String    SourceNodeName,
                      String    SourceMessageName,
                      String    SourceMessageVersion,
                      String    DestinationNodeName,
                      String    DestinationMessageName,
                      String    DestinationMessageVersion);
```

The parameters for this function are described in the following table.

Parameter	Description
InputXMLString InputXMLDoc	This will contain the XML string representation or an XmlDoc object of the input message structure.
AETransformName	Name of the application engine program, of type “Transform Only,” to use for the transform.
SourceNodeName	Source PeopleSoft node name.
SourceMessageName	Source message name to use for the transform.
SourceMessageVersion	Source message version to use for the transform.
DestinationNodeName	Destination PeopleSoft node name.
DestinationMessageName	Destination message name to use for the transform.
DestinationMessageVersion	Destination message version to use for the transform.

The output of this function is an XmlDoc object that contains the final output of the transform.

The follow code shows an example of using the Transform function.

```
/* Create an empty XmlDoc object */
Local XmlDoc &inputDoc = CreateXmlDoc("");

/* Load in an Xml file from disk */
Local boolean &ret = &inputDoc.ParseXmlFromURL
    ("c:\temp\PURCHASE_ORDER.xml");

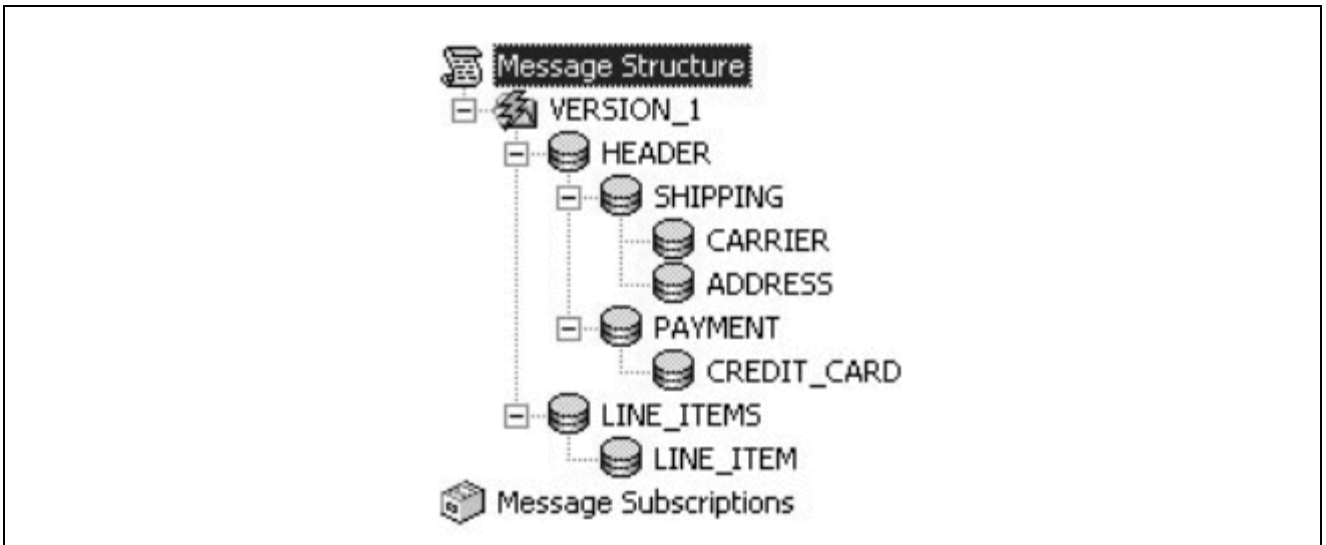
/* Transform the Xml structure */
Local XmlDoc &transformedDoc = Transform(&inputDoc, "PO_TO_ROPO",
    "PSFT_SCM", "PURCHASE_ORDER", "VERSION_1", "PSFT_CRM", "RO_PO",
    "VERSION_1");
```

Transformation Example: Integration Between Two PeopleSoft Nodes

The following example describes 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.

PeopleSoft SCM Node Message Definition

The following PURCHASE_ORDER PeopleSoft message has been created on the PeopleSoft SCM node.



Message PURCHASE_ORDER

A test message was created and the following structure was pulled from Integration Broker Monitor.

```

<?xml version="1.0"?>
<PURCHASE_ORDER>
  <FieldTypes>
    <HEADER class="R">
      <ORDER_NUMBER type="CHAR"></ORDER_NUMBER>
      <ORDER_DATE type="DATE"></ORDER_DATE>
      <TOTAL type="NUMBER"></TOTAL>
      <TAX type="NUMBER"></TAX>
    </HEADER>
    <LINE_ITEMS class="R">
      <NOTE type="CHAR"></NOTE>
    </LINE_ITEMS>
    <SHIPPING class="R">
      <PRICE type="NUMBER"></PRICE>
      <NOTE type="CHAR"></NOTE>
    </SHIPPING>
    <PAYMENT class="R">
      <NOTE type="CHAR"></NOTE>
    </PAYMENT>
  </FieldTypes>
</PURCHASE_ORDER>

```

```

<LINE_ITEM class="R">
  <PART_NUMBER type="CHAR"></PART_NUMBER>
  <DESCRIPTION type="CHAR"></DESCRIPTION>
  <QUANTITY type="NUMBER"></QUANTITY>
  <UOM type="CHAR"></UOM>
  <PRICE type="NUMBER"></PRICE>
  <NOTE type="CHAR"></NOTE>
</LINE_ITEM>
<CREDIT_CARD class="R">
  <TYPE type="CHAR"></TYPE>
  <EXPIRATION_DATE type="DATE"></EXPIRATION_DATE>
  <NAME type="CHAR"></NAME>
  <NUMBER type="NUMBER"></NUMBER>
</CREDIT_CARD>
<CARRIER class="R">
  <NAME type="CHAR"></NAME>
  <TYPE type="CHAR"></TYPE>
  <NOTE type="CHAR"></NOTE>
</CARRIER>
<ADDRESS class="R">
  <NAME type="CHAR"></NAME>
  <ADDRESS1 type="CHAR"></ADDRESS1>
  <ADDRESS2 type="CHAR"></ADDRESS2>
  <CITY type="CHAR"></CITY>
  <STATE type="CHAR"></STATE>
  <ZIP type="CHAR"></ZIP>
</ADDRESS>
<PSCAMA class="R">
  <LANGUAGE_CD type="CHAR"></LANGUAGE_CD>
  <AUDIT_ACTN type="CHAR"></AUDIT_ACTN>
  <BASE_LANGUAGE_CD type="CHAR"></BASE_LANGUAGE_CD>
  <MSG_SEQ_FLG type="CHAR"></MSG_SEQ_FLG>
  <PROCESS_INSTANCE type="NUMBER"></PROCESS_INSTANCE>
  <PUBLISH_RULE_ID type="CHAR"></PUBLISH_RULE_ID>
  <MSGNODENAME type="CHAR"></MSGNODENAME>
</PSCAMA>
</FieldTypes>
<MsgData>
  <Transaction>
    <HEADER class="R">
      <ORDER_NUMBER>A3458D</ORDER_NUMBER>
      <ORDER_DATE>12/05/2001</ORDER_DATE>
      <TOTAL>0</TOTAL>
      <TAX>0.08</TAX>
      <SHIPPING class="R">
        <PRICE>14.06</PRICE>
        <NOTE>Please leave with next door neighbor.</NOTE>
        <CARRIER class="R">
          <NAME>Federal Express</NAME>
          <TYPE>Ground</TYPE>

```

```

        <NOTE></NOTE>
    </CARRIER>
    <ADDRESS class="R">
        <NAME>John Doe</NAME>
        <ADDRESS1>4675 Flying Cloud Terrace</ADDRESS1>
        <ADDRESS2></ADDRESS2>
        <CITY>Stockton</CITY>
        <STATE>CA</STATE>
        <ZIP>95207</ZIP>
    </ADDRESS>
</SHIPPING>
<PAYMENT class="R">
    <NOTE></NOTE>
    <CREDIT_CARD class="R">
        <TYPE>VISA</TYPE>
        <EXPIRATION_DATE>01/04</EXPIRATION_DATE>
        <NAME>John Doe</NAME>
        <NUMBER>4024100293939283</NUMBER>
    </CREDIT_CARD>
</PAYMENT>
</HEADER>
<LINE_ITEMS class="R">
    <NOTE></NOTE>
    <LINE_ITEM class="R">
        <PART_NUMBER>12994</PART_NUMBER>
        <DESCRIPTION>Number 2 pencil</DESCRIPTION>
        <QUANTITY>100</QUANTITY>
        <UOM>Box</UOM>
        <PRICE>0.02</PRICE>
        <NOTE></NOTE>
    </LINE_ITEM>
    <LINE_ITEM class="R">
        <PART_NUMBER>34543</PART_NUMBER>
        <DESCRIPTION>Ultra Soft-Doux Kleenex</DESCRIPTION>
        <QUANTITY>10</QUANTITY>
        <UOM>Carton</UOM>
        <PRICE>1.00</PRICE>
        <NOTE></NOTE>
    </LINE_ITEM>
    <LINE_ITEM class="R">
        <PART_NUMBER>463</PART_NUMBER>
        <DESCRIPTION>Filling cabinet</DESCRIPTION>
        <QUANTITY>2</QUANTITY>
        <UOM>Unit</UOM>
        <PRICE>86.75</PRICE>
        <NOTE></NOTE>
    </LINE_ITEM>
</LINE_ITEMS>
<PSCAMA class="R">
    <LANGUAGE_CD>ENG</LANGUAGE_CD>

```

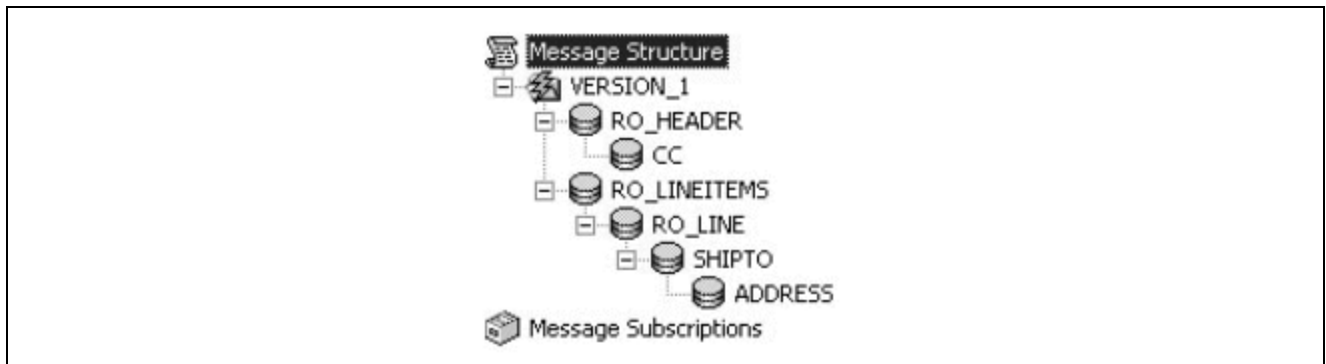
```

        <AUDIT_ACTN></AUDIT_ACTN>
        <BASE_LANGUAGE_CD>ENG</BASE_LANGUAGE_CD>
        <MSG_SEQ_FLG></MSG_SEQ_FLG>
        <PROCESS_INSTANCE>0</PROCESS_INSTANCE>
        <PUBLISH_RULE_ID></PUBLISH_RULE_ID>
        <MSGNODENAME></MSGNODENAME>
    </PSCAMA>
</Transaction>
</MsgData>
</PURCHASE_ORDER>

```

PeopleSoft CRM Node Message Definition

The following PeopleSoft message RO_PO has been created on the PeopleSoft CRM node.



Message RO_PO

A test message was created and the following structure was pulled from Integration Broker Monitor.

```

<?xml version="1.0"?>
<RO_PO>
  <FieldTypes>
    <RO_HEADER class="R">
      <ORDNUM type="CHAR"></ORDNUM>
      <ORDDATE type="DATE"></ORDDATE>
      <TOTAL type="NUMBER"></TOTAL>
      <TAX type="NUMBER"></TAX>
    </RO_HEADER>
    <RO_LINEITEMS class="R">
      <NOTE type="CHAR"></NOTE>
    </RO_LINEITEMS>
    <CC class="R">
      <EXDATE type="DATE"></EXDATE>
      <AE type="CHAR"></AE>
      <VISA type="CHAR"></VISA>
      <MC type="CHAR"></MC>
      <NAME type="CHAR"></NAME>
    </CC>
    <SHIPTO class="R">
      <NAME type="CHAR"></NAME>
    </SHIPTO>
  </FieldTypes>
</RO_PO>

```



```

        <PRICE type="NUMBER"></PRICE>
    </SHIPTO>
    <RO_LINE class="R">
        <ORDNUM type="CHAR"></ORDNUM>
        <PARTNUM type="CHAR"></PARTNUM>
        <QTY type="NUMBER"></QTY>
        <PRICE type="NUMBER"></PRICE>
        <UNITOFMEASURE type="CHAR"></UNITOFMEASURE>
        <DESCR type="CHAR"></DESCR>
    </RO_LINE>
    <ADDRESS class="R">
        <NAME type="CHAR"></NAME>
        <ZIPCODE type="CHAR"></ZIPCODE>
        <ADDRESS1 type="CHAR"></ADDRESS1>
        <ADDRESS2 type="CHAR"></ADDRESS2>
        <CITY type="CHAR"></CITY>
        <STATE type="CHAR"></STATE>
    </ADDRESS>
    <PSCAMA class="R">
        <LANGUAGE_CD type="CHAR"></LANGUAGE_CD>
        <AUDIT_ACTN type="CHAR"></AUDIT_ACTN>
        <BASE_LANGUAGE_CD type="CHAR"></BASE_LANGUAGE_CD>
        <MSG_SEQ_FLG type="CHAR"></MSG_SEQ_FLG>
        <PROCESS_INSTANCE type="NUMBER"></PROCESS_INSTANCE>
        <PUBLISH_RULE_ID type="CHAR"></PUBLISH_RULE_ID>
        <MSGNODENAME type="CHAR"></MSGNODENAME>
    </PSCAMA>
</FieldTypes>
<MsgData>
    <Transaction>
        <RO_HEADER class="R">
            <ORDNUM/>
            <ORDDATE/>
            <TOTAL/>
            <TAX/>
            <CC class="R">
                <EXDATE/>
                <AE/>
                <VISA/>
                <MC/>
                <NAME/>
            </CC>
        </RO_HEADER>
        <RO_LINEITEMS class="R">
            <NOTE/>
            <RO_LINE class="R">
                <ORDNUM/>
                <PARTNUM/>
                <QTY/>
                <PRICE/>
            </RO_LINE>
        </RO_LINEITEMS>
    </Transaction>
</MsgData>

```

```

        <UNITOFMEASURE/>
        <DESCR/>
        <SHIPTO class="R">
            <NAME></NAME>
            <PRICE></PRICE>
            <ADDRESS class="R">
                <NAME/>
                <ZIPCODE/>
                <ADDRESS1/>
                <ADDRESS2/>
                <CITY/>
                <STATE/>
            </ADDRESS>
        </SHIPTO>
    </RO_LINE>
</RO_LINEITEMS>
<PSCAMA class="R">
    <LANGUAGE_CD>ENG</LANGUAGE_CD>
    <AUDIT_ACTN/>
    <BASE_LANGUAGE_CD>ENG</BASE_LANGUAGE_CD>
    <MSG_SEQ_FLG/>
    <PROCESS_INSTANCE/>
    <PUBLISH_RULE_ID/>
    <MSGNODENAME/>
</PSCAMA>
</Transaction>
</MsgData>
</RO_PO>

```

Setting Up the Transformation

This section describes setting up a transformation for the PeopleSoft SCM to PeopleSoft CRM integration.

PeopleSoft SCM to PeopleSoft CRM Integration

A new application engine program was created with one XSLT step containing the following:

```

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

    <xsl:template match="PURCHASE_ORDER">
        <RO_PO>
            <xsl:apply-templates/>
        </RO_PO>
    </xsl:template>

    <xsl:template match="HEADER">
        <RO_HEADER><xsl:attribute name="class"><xsl:value-of select=
"@class"/></xsl:attribute>
            <ORDNUM><xsl:value-of select="ORDER_NUMBER"/></ORDNUM>
        </RO_HEADER>
    </xsl:template>

```

```

        <ORDDATE><xsl:value-of select="ORDER_DATE"/></ORDDATE>
        <TOTAL><xsl:value-of select="TOTAL"/></TOTAL>
        <TAX><xsl:value-of select="TAX"/></TAX>

        <!-- Get the credit card info if present -->
        <xsl:apply-templates select="PAYMENT/CREDIT_CARD"/>

    </RO_HEADER>
</xsl:template>

<xsl:template match="CREDIT_CARD">
    <CC><xsl:attribute name="class"><xsl:value-of select="@class"/>
        </xsl:attribute>
        <EXDATE><xsl:value-of select="EXPIRATION_DATE"/></EXDATE>
        <psft_function name="codeset" codesetname="BUILD_CC">
            <parm name="RO_CCTYPE"><xsl:value-of select="TYPE"/>
            </parm>
            <value name="CARDTYPE" select="." createNodeFromValue=
                "yes"><xsl:value-of select="NUMBER"/></value>
        </psft_function>
        <NAME><xsl:value-of select="NAME"/></NAME>
    </CC>
</xsl:template>

<xsl:template match="LINE_ITEMS">

    <!-- Get the Order number -->
    <xsl:param name="orderNumber"><xsl:value-of select="..
        /HEADER/ORDER_NUMBER"/></xsl:param>

    <RO_LINEITEMS><xsl:attribute name="class"><xsl:value-of
        select="@class"/></xsl:attribute>
        <NOTE><xsl:value-of select="NOTE"/></NOTE>
        <ORDNUM><xsl:value-of select="$orderNumber"/></ORDNUM>

    <!-- Apply the templates, make sure you pass in the ordernumber -->
    <xsl:apply-templates>
        <xsl:with-param name="orderNumber"><xsl:value-of select=
            "$orderNumber"/></xsl:with-param>
    </xsl:apply-templates>
    </RO_LINEITEMS>
</xsl:template>

<xsl:template match="LINE_ITEM">

    <!-- Define the passed parameters -->
    <xsl:param name="orderNumber"/>

    <RO_LINE><xsl:attribute name="class"><xsl:value-of select="@class"/>
        </xsl:attribute>

```

```

    <ORDNUM><xsl:value-of select="$orderNumber"/></ORDNUM>
    <PARTNUM><xsl:value-of select="PART_NUMBER"/></PARTNUM>
    <QTY><xsl:value-of select="QUANTITY"/></QTY>
    <UNITOFMEASURE>
      <psft_function name="codeset" codesetname="RO_UOM_CHNG">
        <parm name="RO_UOM"><xsl:value-of select="UOM"/></parm>
        <value name="UOM" select="."/>
      </psft_function>
    </UNITOFMEASURE>
    <DESCR><xsl:value-of select="DESCRIPTION"/></DESCR>

    <!-- Pick up the rest of the nodes and disregard the others -->
    <xsl:apply-templates/>

    <!-- Get the shipping address -->
    <xsl:apply-templates select="../../HEADER/SHIPPING"/>

  </RO_LINE>
</xsl:template>

<xsl:template match="SHIPPING">
  <SHIPTO class="R">
    <NAME>
      <psft_function name="codeset" codesetname="RO_CARRIER">
        <parm name="RO_CARRIER"><xsl:value-of select=
          "CARRIER/NAME"/></parm>
        <parm name="RO_CARRIER_TYPE"><xsl:value-of select=
          "CARRIER/TYPE"/></parm>
        <value name="CARRIER" select="."/>
      </psft_function>
    </NAME>
    <xsl:apply-templates/>
  </SHIPTO>
</xsl:template>

<xsl:template match="ZIP">
  <ZIPCODE><xsl:value-of select="."/></ZIPCODE>
</xsl:template>

<!-- This will copy a node/attribute and it's children -->
<xsl:template match="node() | @*">
  <xsl:copy>
    <xsl:apply-templates select="node() | @*" />
  </xsl:copy>
</xsl:template>

<!-- This will not copy these nodes -->
<xsl:template match="processing-instruction() | NOTE | CARRIER
  | PART_NUMBER | DESCRIPTION | QUANTITY | UOM">
  <!-- Do nothing when you find these nodes -->

```

```

</xsl:template>

<xsl:template match="FieldTypes">
  <FieldTypes>
    <RO_HEADER class="R">
      <ORDNUM type="CHAR"/>
      <ORDDATE type="DATE"/>
      <TOTAL type="NUMBER"/>
      <TAX type="NUMBER"/>
    </RO_HEADER>
    <RO_LINEITEMS class="R">
      <NOTE type="CHAR"/>
    </RO_LINEITEMS>
    <CC class="R">
      <EXDATE type="DATE"/>
      <AE type="CHAR"/>
      <VISA type="CHAR"/>
      <MC type="CHAR"/>
      <NAME type="CHAR"/>
    </CC>
    <SHIPTO class="R">
      <NAME type="CHAR"/>
      <PRICE type="NUMBER"/>
    </SHIPTO>
    <RO_LINE class="R">
      <ORDNUM type="CHAR"/>
      <PARTNUM type="CHAR"/>
      <QTY type="NUMBER"/>
      <PRICE type="NUMBER"/>
      <UNITOFMEASURE type="CHAR"/>
      <DESCR type="CHAR"/>
    </RO_LINE>
    <ADDRESS class="R">
      <NAME type="CHAR"/>
      <ZIPCODE type="CHAR"/>
      <ADDRESS1 type="CHAR"/>
      <ADDRESS2 type="CHAR"/>
      <CITY type="CHAR"/>
      <STATE type="CHAR"/>
    </ADDRESS>
    <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>

```

XSLT Template Walkthrough

An XSLT template is an instruction of what to do when a particular XML node/condition is encountered.

The following template is set to run logic when an XML node named *PURCHASE_ORDER* is encountered. It states that it will output a node named *RO_PO* and continue on to call its sibling templates in the XSLT.

```

<xsl:template match="PURCHASE_ORDER">
    <RO_PO>
        <xsl:apply-templates/>
    </RO_PO>
</xsl:template>

```

The root node of each XML Document that passes through this transformation will be *PURCHASE_ORDER*. When the input Document is consumed the root node will be the first one processed. This means that the first template to be processed will be the *PURCHASE_ORDER* template. One can see that once this template is called, an *RO_PO* node is created. After the node is created, the transformation engine is told to apply templates to each child node found in the *PURCHASE_ORDER* root node.

The first child node the transformation engine will encounter will be the FieldTypes node. The FieldTypes node will call the following template:

```

<xsl:template match="FieldTypes">
    <FieldTypes>
        <RO_HEADER class="R">
            <ORDNUM type="CHAR"/>
            <ORDDATE type="DATE"/>
            <TOTAL type="NUMBER"/>
            <TAX type="NUMBER"/>
        </RO_HEADER>
        <RO_LINEITEMS class="R">
            <NOTE type="CHAR"/>
        </RO_LINEITEMS>
        <CC class="R">
            <EXDATE type="DATE"/>
            <AE type="CHAR"/>
            <VISA type="CHAR"/>
            <MC type="CHAR"/>
            <NAME type="CHAR"/>
        </CC>
        <SHIPTO class="R">
            <NAME type="CHAR"/>
            <PRICE type="NUMBER"/>
        </SHIPTO>
        <RO_LINE class="R">
            <ORDNUM type="CHAR"/>
            <PARTNUM type="CHAR"/>
            <QTY type="NUMBER"/>

```

```

        <PRICE type="NUMBER"/>
        <UNITOFMEASURE type="CHAR"/>
        <DESCR type="CHAR"/>
    </RO_LINE>
    <ADDRESS class="R">
        <NAME type="CHAR"/>
        <ZIPCODE type="CHAR"/>
        <ADDRESS1 type="CHAR"/>
        <ADDRESS2 type="CHAR"/>
        <CITY type="CHAR"/>
        <STATE type="CHAR"/>
    </ADDRESS>
    <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>

```

This is a rather simple template; its main purpose is to output the field type information needed for the RO_PO PeopleSoft Message. The transformation engine will then encounter the MsgData node. Instead of having a template explicitly defined to handle a MsgData node, it would be extremely nice to have a simple generic copy template. When using this copy functionality, make sure to have a template defined that will ignore the processing-instruction() node.

```

<!-- This will copy a node/attribute and it's children -->
<xsl:template match="node() | @">
    <xsl:copy>
        <xsl:apply-templates select="node() | @"/>
    </xsl:copy>
</xsl:template>

<!-- Do not copy -->
<xsl:template match="processing-instruction()">
    <!-- Do nothing -->
</xsl:template>

```

When the transformation engine encounters the MsgData node and does not find a corresponding MsgData template, it will call the above template. This template states the following, when a node or attribute is found, copy it and then continue to apply templates to its children.

This means that when the MsgData node is encountered, it will be copied to the output and processing will continue on to the Transaction child node. Again, there is no template defined for this node, so the above template will be called. As the transformation engine continues to walk down the XML tree structure it will encounter the Header node.

The following template has been defined to handle a Header node, which is required to output the desired RO_HEADER node contained in the RO_PO message:

```
<xsl:template match="HEADER">
  <RO_HEADER><xsl:attribute name="class"><xsl:value-of select
    ="@class"/></xsl:attribute>
    <ORDNUM><xsl:value-of select="ORDER_NUMBER"/></ORDNUM>
    <ORDDATE><xsl:value-of select="ORDER_DATE"/></ORDDATE>
    <TOTAL><xsl:value-of select="TOTAL"/></TOTAL>
    <TAX><xsl:value-of select="TAX"/></TAX>

    <!-- Get the credit card info if present -->
    <xsl:apply-templates select="PAYMENT/CREDIT_CARD"/>

  </RO_HEADER>
</xsl:template>
```

Always remember that the input drives this recursive process, so each XSLT must be created with this in mind. Since the input XML document contains a Header node, you must create a template for it. Once this template is called the transformation engine begins executing from the perspective of the Header node. All XPath's contained within this template must specify a path relative to the Header node location. For example, look at the following snippet:

```
<xsl:attribute name="class"><xsl:value-of select="@class"/></xsl:attribute>
```

The transformation engine is being told to create an attribute with a name of class and its value should be created from the class attribute found in the Header node. The xsl:value-of is used to accomplish this. Given an XPath, the xsl:value-of node will navigate to the specified path, find the value and replace itself with the found value. If no value is found, the xsl:value-of node will simply be removed.

```
<ORDNUM><xsl:value-of select="ORDER_NUMBER"/></ORDNUM>
  <ORDDATE><xsl:value-of select="ORDER_DATE"/></ORDDATE>
  <TOTAL><xsl:value-of select="TOTAL"/></TOTAL>
  <TAX><xsl:value-of select="TAX"/></TAX>
```

These statements are doing an example of a simple mapping, which basically means an XML node name change. The ORDER_NUMBER node found in the input Document is being changed to ORDNUM. The final action of this template is to find credit card information.

```
<!-- Get the credit card info if present -->
  <xsl:apply-templates select="PAYMENT/CREDIT_CARD"/>
```

This is a very specific search here, because only credit card information is required at this output location. Instead of telling the transformation engine to continue applying templates to all further child nodes, one specific node has been chosen. This has been accomplished by specifying the relative XPath in the select attribute. The XPath that has been specified tells the transformation engine to look for a child node of PAYMENT which has a child node of CREDIT_CARD.

A CREDIT_CARD template does indeed exist, so the following logic will be executed:

```
<xsl:template match="CREDIT_CARD">
  <CC><xsl:attribute name="class"><xsl:value-of select="@class"/>
    </xsl:attribute>
    <EXDATE><xsl:value-of select="EXPIRATION_DATE"/></EXDATE>
    <psft_function name="codeset" codesetname=" BUILD_CC">
```



```

        <parm name="RO_CCTYPE"><xsl:value-of select="TYPE"/></parm>
        <value name="CARDTYPE" select="."
        createNodeFromValue="yes">
        <xsl:value-of select="NUMBER"/>
        </value></psft_function>
        <NAME><xsl:value-of select="NAME"/></NAME>
    </CC>
</xsl:template>

```

The creation of the class attribute and a few mappings between differently named nodes occurs. What makes this template different though, is that a psft_function node exists. This means that a codeset lookup is required at this point. The following codeset set up has been completed

From Group: PSFT_SCM
Code Set Name: BUILD_CC
To Group: PSFT_CRM

Code Sets Find | View All First 1 of 3 Last

Description: Visa + -

Select	Match Name	Match Value
☐	RO_CCTYPE	American Express
☐	RO_CCTYPE	Discover
☐	RO_CCTYPE	Master Card
☒	RO_CCTYPE	VISA

Code Set Values Customize | Find | View All First 1 of 1 Last

*Return Name	Return Value
CARDTYPE	VISA

Codeset values for RO_CCTYPE

The rest of the possible RO_CCTYPE values have been setup as well, but are not useful to this example.

Once the CREDIT_CARD template has finished executing, the transformation engine that previously had been told to apply templates to all of the nodes found at the Header node level continues on to encounter the LINE_ITEMS node. This illustrates a classic example of how recursion works, by processing down the entire length of one branch and then bubbling back up to process down yet another branch of the tree. When the LINE_ITEMS node is encountered, the following logic will be executed:

```

<xsl:template match="LINE_ITEMS">

    <!-- Get the Order number -->
    <xsl:param name="orderNumber">
        <xsl:value-of select="../HEADER/ORDER_NUMBER"/></xsl:param>

    <RO_LINEITEMS><xsl:attribute name="class"><xsl:value-of select
    ="@class"/></xsl:attribute>
        <NOTE><xsl:value-of select="NOTE"/></NOTE>

```

```

        <ORDNUM><xsl:value-of select="$orderNumber"/>
    </ORDNUM>

    <!-- Apply the templates, make sure you pass in the ordernumber -->
    <xsl:apply-templates>
        <xsl:with-param name="orderNumber">
            <xsl:value-of select="$orderNumber"/></xsl:with-param>
        </xsl:apply-templates>
    </RO_LINEITEMS>
</xsl:template>

```

Contained within this template is an example of creating parameters and passing those parameters to different templates. A parameter has been created with the name of *orderNumber* which will be allocated a value from the input Document. The XPath *../HEADER/ORDER_NUMBER* indicates the desired value's location relative to the *LINE_ITEMS* node. This XPath states to back up to the parent node of *LINE_ITEMS*, find a child node *HEADER* and then retrieve the value from its child node *ORDER_NUMBER*.

This parameter will be used as the output value for the *ORDNUM* node.

When a parameter is selected, it must be preceded by the \$ character. When this character is encountered in a select, the transformation engine expects a parameter name to follow. An error will be thrown if the expected parameter name is not found. Also, parameters fall out of scope when leaving a template; they can only be preserved if they are programmatically passed into other templates, or declared globally.

The *LINE_ITEMS* template has finished processing and it has told the transformation engine to apply templates to its child nodes. Instead of simply apply templates this time; the developer has decided to preserve the *orderNumber* parameter by passing it along.

The next group of nodes encountered will be the *LINE_ITEM* nodes, which will use the following template for each *LINE_ITEM* node found at this level:

```

<xsl:template match="LINE_ITEM">
    <!-- Define the passed parameters -->
    <xsl:param name="orderNumber"/>

    <RO_LINE><xsl:attribute name="class"><xsl:value-of select="@class"/>
        </xsl:attribute>
        <ORDNUM><xsl:value-of select="$orderNumber"/></ORDNUM>
        <PARTNUM><xsl:value-of select="PART_NUMBER"/></PARTNUM>
        <QTY><xsl:value-of select="QUANTITY"/></QTY>
        <UNITOFMEASURE>
            <psft_function name="codeset" codesetname="RO_UOM_CHNG">
                <parm name="RO_UOM"><xsl:value-of select="UOM"/></parm>
                <value name="UOM" select="."/>
            </psft_function>
        </UNITOFMEASURE>
        <DESCR><xsl:value-of select="DESCRIPTION"/></DESCR>

    <!-- Pick up the rest of the nodes and disregard the others -->
    <xsl:apply-templates/>
    <!-- Get the shipping address -->
    <xsl:apply-templates select="../../HEADER/SHIPPING"/>

```

```

    </RO_LINE>
</xsl:template>

```

The LINE_ITEM template is expecting the orderNumber parameter; this is why the parameter is defined. If a template does not care about a parameter, it should not be defined.

Remember, if there is a specific set of logic that needs to happen when a node is encountered, create a template for that node name. On the other hand, if nodes simply need to be copied, make sure the following template exists in the XSLT:

```

<!-- This will copy a node/attribute and it's children -->
<xsl:template match="node() | @">
  <xsl:copy>
    <xsl:apply-templates select="node() | @"/>
  </xsl:copy>
</xsl:template>

<!-- Do not copy -->
<xsl:template match="processing-instruction()">
  <!-- Do nothing -->
</xsl:template>

```

This will make sure to copy all nodes and their children, as well as tell the transformation engine to apply templates to nodes with corresponding templates if they are found. This means that if the transformation engine encounters the LINE_ITEMS node, it will call a LINE_ITEMS template rather than the above generic copying template.

The transformation engine can also be told to do nothing when a node is encountered. This can be accomplished with something similar to the following:

```

<!-- This will not copy these nodes -->
<xsl:template match="NOTE | CARRIER | PART_NUMBER | DESCRIPTION
  | QUANTITY | UOM">
  <!-- Do nothing when you find these nodes -->
</xsl:template>

```

This template has been set to catch the following nodes: NOTE, CARRIER, PART_NUMBER, DESCRIPTION, QUANTITY and UOM. In some instances you might want to do a simple copy of a node and its children, but filter out a few of those child nodes. Instead of writing code to select each individual node desired in the output, you can create a template which performs a “do nothing” to simply eliminate (filter out) certain nodes when they are encountered.

Associating the Transformation to a Transaction

Up to this point it has been assumed that the proper relationship setup has been completed. To complete this entire process, make sure to set the transformation to fire when publishing the outbound request for the desired transaction.

Relationship ID:	SCM_TO_CRM		
Nodes:	PSFT_CRM	PSFT_CRM	
Transaction Modifiers			
Effective Date:	12/21/2001	*Status:	Active v
		Sequence number:	
Initial			
Node:	PSFT_CRM	Transaction Type:	OA
Rqst Msg:	PURCHASE_ORDER	Version:	VERSION_1
Result			
Node:	PSFT_CRM	*Transaction Type:	OA v
Rqst Msg:	PURCHASE_ORDER	Version:	VERSION_1
Transformations			
Request:	PO_TO_ROPO Q		

Associating a transformation to a transaction

Understanding the Output After the First XSLT Pass

The following code example show the output after the first XSLT pass. Note that the output contains the psft_function nodes that will be consumed to process codeset functionality on the second pass.

```
<?xml version="1.0"?>
<RO_PO>
  <FieldTypes>
    <RO_HEADER class="R">
      <ORDNUM type="CHAR"></ORDNUM>
      <ORDDATE type="DATE"></ORDDATE>
      <TOTAL type="NUMBER"></TOTAL>
      <TAX type="NUMBER"></TAX>
    </RO_HEADER>
    <RO_LINEITEMS class="R">
      <NOTE type="CHAR"></NOTE>
    </RO_LINEITEMS>
    <CC class="R">
      <EXDATE type="DATE"></EXDATE>
      <AE type="CHAR"></AE>
      <VISA type="CHAR"></VISA>
      <MC type="CHAR"></MC>
      <NAME type="CHAR"></NAME>
    </CC>
    <SHIPTO class="R">
      <NAME type="CHAR"></NAME>
      <PRICE type="NUMBER"></PRICE>
    </SHIPTO>
  </RO_PO>
  <RO_LINE class="R">
```

```

        <ORDNUM type="CHAR"></ORDNUM>
        <PARTNUM type="CHAR"></PARTNUM>
        <QTY type="NUMBER"></QTY>
        <PRICE type="NUMBER"></PRICE>
        <UNITOFMEASURE type="CHAR"></UNITOFMEASURE>
        <DESCR type="CHAR"></DESCR>
    </RO_LINE>
    <ADDRESS class="R">
        <NAME type="CHAR"></NAME>
        <ZIPCODE type="CHAR"></ZIPCODE>
        <ADDRESS1 type="CHAR"></ADDRESS1>
        <ADDRESS2 type="CHAR"></ADDRESS2>
        <CITY type="CHAR"></CITY>
        <STATE type="CHAR"></STATE>
    </ADDRESS>
    <PSCAMA class="R">
        <LANGUAGE_CD type="CHAR"></LANGUAGE_CD>
        <AUDIT_ACTN type="CHAR"></AUDIT_ACTN>
        <BASE_LANGUAGE_CD type="CHAR"></BASE_LANGUAGE_CD>
        <MSG_SEQ_FLG type="CHAR"></MSG_SEQ_FLG>
        <PROCESS_INSTANCE type="NUMBER"></PROCESS_INSTANCE>
        <PUBLISH_RULE_ID type="CHAR"></PUBLISH_RULE_ID>
        <MSGNODENAME type="CHAR"></MSGNODENAME>
    </PSCAMA>
</FieldTypes>
<MsgData>
    <Transaction>
        <RO_HEADER class="R">
            <ORDNUM>A3458D</ORDNUM>
            <ORDDATE>12/05/2001</ORDDATE>
            <TOTAL>0</TOTAL>
            <TAX>0.08</TAX>
            <CC class="R">
                <EXDATE>01/04</EXDATE>
                <psft_function name="codeset" codesetname="BUILD_CC">
                    <parm name="RO_CCTYPE">VISA</parm>
                    <value name="CARDTYPE" select="."
                        createNodeFromValue="yes">4024100293939283
                    </value>
                </psft_function>
                <NAME>John Doe</NAME>
            </CC>
        </RO_HEADER>
        <RO_LINEITEMS class="R">
            <NOTE></NOTE>
            <ORDNUM>A3458D</ORDNUM>
            <RO_LINE class="R">
                <ORDNUM>A3458D</ORDNUM>
                <PARTNUM>12994</PARTNUM>
                <QTY>100</QTY>
            </RO_LINE>
        </RO_LINEITEMS>
    </Transaction>
</MsgData>

```

```

<UNITOFMEASURE>
  <psft_function name="codeset" codesetname=
    "RO_UOM_CHNG">
    <parm name="RO_UOM">Box</parm>
    <value name="UOM" select="."/>
  </psft_function>
</UNITOFMEASURE>
<DESCR>Number 2 pencil</DESCR>
<PRICE>0.02</PRICE>
<SHIPTO class="R">
  <NAME>
    <psft_function name="codeset" codesetname=
      "RO_CARRIER">
      <parm name="RO_CARRIER">
        Federal Express
      </parm>
      <parm name="RO_CARRIER_TYPE">
        Ground
      </parm>
      <value name="CARRIER" select="."/>
    </psft_function>
  </NAME>
  <PRICE>14.06</PRICE>
  <ADDRESS class="R">
    <NAME>John Doe</NAME>
    <ADDRESS1>
      4675 Flying Cloud Terrace
    </ADDRESS1>
    <ADDRESS2></ADDRESS2>
    <CITY>Stockton</CITY>
    <STATE>CA</STATE>
    <ZIPCODE>95207</ZIPCODE>
  </ADDRESS>
</SHIPTO>
</RO_LINE>
<RO_LINE class="R">
  <ORDNUM>A3458D</ORDNUM>
  <PARTNUM>34543</PARTNUM>
  <QTY>10</QTY>
  <UNITOFMEASURE>
    <psft_function name="codeset" codesetname=
      "RO_UOM_CHNG">
      <parm name="RO_UOM">Carton</parm>
      <value name="UOM" select="."/>
    </psft_function>
  </UNITOFMEASURE>
  <DESCR>Ultra Soft-Doux Kleenex</DESCR>
  <PRICE>1.00</PRICE>
  <SHIPTO class="R">
    <NAME>

```

```

        <psft_function name="codeset" codesetname=
        "RO_CARRIER">
            <parm name="RO_CARRIER">
                Federal Express
            </parm>
            <parm name="RO_CARRIER_TYPE">
                Ground</parm>
            <value name="CARRIER" select="."/>
        </psft_function>
    </NAME>
    <PRICE>14.06</PRICE>
    <ADDRESS class="R">
        <NAME>John Doe</NAME>
        <ADDRESS1>4675 Flying Cloud Terrace</ADDRESS1>
        <ADDRESS2></ADDRESS2>
        <CITY>Stockton</CITY>
        <STATE>CA</STATE>
        <ZIPCODE>95207</ZIPCODE>
    </ADDRESS>
    </SHIPTO>
</RO_LINE>
<RO_LINE class="R">
    <ORDNUM>A3458D</ORDNUM>
    <PARTNUM>463</PARTNUM>
    <QTY>2</QTY>
    <UNITOFMEASURE>
        <psft_function name="codeset" codesetname=
        "RO_UOM_CHNG">
            <parm name="RO_UOM">Unit</parm>
            <value name="UOM" select="."/>
        </psft_function>
    </UNITOFMEASURE>
    <DESCR>Filling cabinet</DESCR>
    <PRICE>86.75</PRICE>
    <SHIPTO class="R">
        <NAME>
            <psft_function name="codeset" codesetname=
            "RO_CARRIER">
                <parm name="RO_CARRIER">
                    Federal Express
                </parm>
                <parm name="RO_CARRIER_TYPE">
                    Ground
                </parm>
                <value name="CARRIER" select="."/>
            </psft_function>
        </NAME>
        <PRICE>14.06</PRICE>
        <ADDRESS class="R">
            <NAME>John Doe</NAME>

```

```

        <ADDRESS1>4675 Flying Cloud Terrace</ADDRESS1>
        <ADDRESS2></ADDRESS2>
        <CITY>Stockton</CITY>
        <STATE>CA</STATE>
        <ZIPCODE>95207</ZIPCODE>
    </ADDRESS>
</SHIPTO>
</RO_LINE>
</RO_LINEITEMS>
<PSCAMA class="R">
    <LANGUAGE_CD>ENG</LANGUAGE_CD>
    <AUDIT_ACTN></AUDIT_ACTN>
    <BASE_LANGUAGE_CD>ENG</BASE_LANGUAGE_CD>
    <MSG_SEQ_FLG></MSG_SEQ_FLG>
    <PROCESS_INSTANCE>0</PROCESS_INSTANCE>
    <PUBLISH_RULE_ID></PUBLISH_RULE_ID>
    <MSGNODENAME></MSGNODENAME>
</PSCAMA>
</Transaction>
</MsgData>
</RO_PO>

```

Understanding the Output After the Second Codeset Pass

The following example show the output after the second codeset pass.

```

<?xml version="1.0"?>
<RO_PO>
  <FieldTypes>
    <RO_HEADER class="R">
      <ORDNUM type="CHAR"></ORDNUM>
      <ORDDATE type="DATE"></ORDDATE>
      <TOTAL type="NUMBER"></TOTAL>
      <TAX type="NUMBER"></TAX>
    </RO_HEADER>
    <RO_LINEITEMS class="R">
      <NOTE type="CHAR"></NOTE>
    </RO_LINEITEMS>
    <CC class="R">
      <EXDATE type="DATE"></EXDATE>
      <AE type="CHAR"></AE>
      <VISA type="CHAR"></VISA>
      <MC type="CHAR"></MC>
      <NAME type="CHAR"></NAME>
    </CC>
    <SHIPTO class="R">
      <NAME type="CHAR"></NAME>
      <PRICE type="NUMBER"></PRICE>
    </SHIPTO>
  </FieldTypes>

```



```

<RO_LINE class="R">
  <ORDNUM type="CHAR"></ORDNUM>
  <PARTNUM type="CHAR"></PARTNUM>
  <QTY type="NUMBER"></QTY>
  <PRICE type="NUMBER"></PRICE>
  <UNITOFMEASURE type="CHAR"></UNITOFMEASURE>
  <DESCR type="CHAR"></DESCR>
</RO_LINE>
<ADDRESS class="R">
  <NAME type="CHAR"></NAME>
  <ZIPCODE type="CHAR"></ZIPCODE>
  <ADDRESS1 type="CHAR"></ADDRESS1>
  <ADDRESS2 type="CHAR"></ADDRESS2>
  <CITY type="CHAR"></CITY>
  <STATE type="CHAR"></STATE>
</ADDRESS>
<PSCAMA class="R">
  <LANGUAGE_CD type="CHAR"></LANGUAGE_CD>
  <AUDIT_ACTN type="CHAR"></AUDIT_ACTN>
  <BASE_LANGUAGE_CD type="CHAR"></BASE_LANGUAGE_CD>
  <MSG_SEQ_FLG type="CHAR"></MSG_SEQ_FLG>
  <PROCESS_INSTANCE type="NUMBER"></PROCESS_INSTANCE>
  <PUBLISH_RULE_ID type="CHAR"></PUBLISH_RULE_ID>
  <MSGNODENAME type="CHAR"></MSGNODENAME>
</PSCAMA>
</FieldTypes>
<MsgData>
  <Transaction>
    <RO_HEADER class="R">
      <ORDNUM>A3458D</ORDNUM>
      <ORDDATE>12/05/2001</ORDDATE>
      <TOTAL>0</TOTAL>
      <TAX>0.08</TAX>
      <CC class="R">
        <EXDATE>01/04</EXDATE>
        <VISA>4024100293939283</VISA>
        <NAME>John Doe</NAME>
      </CC>
    </RO_HEADER>
    <RO_LINEITEMS class="R">
      <NOTE></NOTE>
      <ORDNUM>A3458D</ORDNUM>
      <RO_LINE class="R">
        <ORDNUM>A3458D</ORDNUM>
        <PARTNUM>12994</PARTNUM>
        <QTY>100</QTY>
        <UNITOFMEASURE>B</UNITOFMEASURE>
        <DESCR>Number 2 pencil</DESCR>
        <PRICE>0.02</PRICE>
        <SHIPTO class="R">

```

```

        <NAME>FedEx - Ground</NAME>
        <PRICE>14.06</PRICE>
        <ADDRESS class="R">
            <NAME>John Doe</NAME>
            <ADDRESS1>4675 Flying Cloud Terrace</ADDRESS1>
            <ADDRESS2></ADDRESS2>
            <CITY>Stockton</CITY>
            <STATE>CA</STATE>
            <ZIPCODE>95207</ZIPCODE>
        </ADDRESS>
    </SHIPTO>
</RO_LINE>
<RO_LINE class="R">
    <ORDNUM>A3458D</ORDNUM>
    <PARTNUM>34543</PARTNUM>
    <QTY>10</QTY>
    <UNITOFMEASURE>C</UNITOFMEASURE>
    <DESCR>Ultra Soft-Doux Kleenex</DESCR>
    <PRICE>1.00</PRICE>
    <SHIPTO class="R">
        <NAME>FedEx - Ground</NAME>
        <PRICE>14.06</PRICE>
        <ADDRESS class="R">
            <NAME>John Doe</NAME>
            <ADDRESS1>4675 Flying Cloud Terrace</ADDRESS1>
            <ADDRESS2></ADDRESS2>
            <CITY>Stockton</CITY>
            <STATE>CA</STATE>
            <ZIPCODE>95207</ZIPCODE>
        </ADDRESS>
    </SHIPTO>
</RO_LINE>
<RO_LINE class="R">
    <ORDNUM>A3458D</ORDNUM>
    <PARTNUM>463</PARTNUM>
    <QTY>2</QTY>
    <UNITOFMEASURE>SU</UNITOFMEASURE>
    <DESCR>Filling cabinet</DESCR>
    <PRICE>86.75</PRICE>
    <SHIPTO class="R">
        <NAME>FedEx - Ground</NAME>
        <PRICE>14.06</PRICE>
        <ADDRESS class="R">
            <NAME>John Doe</NAME>
            <ADDRESS1>4675 Flying Cloud Terrace</ADDRESS1>
            <ADDRESS2></ADDRESS2>
            <CITY>Stockton</CITY>
            <STATE>CA</STATE>
            <ZIPCODE>95207</ZIPCODE>
        </ADDRESS>
    </SHIPTO>
</RO_LINE>

```

```
        </SHIPTO>
    </RO_LINE>
</RO_LINEITEMS>
<PSCAMA class="R">
    <LANGUAGE_CD>ENG</LANGUAGE_CD>
    <AUDIT_ACTN></AUDIT_ACTN>
    <BASE_LANGUAGE_CD>ENG</BASE_LANGUAGE_CD>
    <MSG_SEQ_FLG></MSG_SEQ_FLG>
    <PROCESS_INSTANCE>0</PROCESS_INSTANCE>
    <PUBLISH_RULE_ID></PUBLISH_RULE_ID>
    <MSGNODENAME></MSGNODENAME>
</PSCAMA>
</Transaction>
</MsgData>
</RO_PO>
```


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), Post Office Protocol 3 (POP3), and Simple Mail Transfer Protocol (SMTP) communication formats. You can use the PeopleSoft Integration Broker Connector SDK to build and implement connectors for other communication formats and application requirements.

For example, you could develop a connector for integrations between a PeopleSoft Human Resources system and an SAP Financials system. You could also develop a connector for integrations between a PeopleSoft Supply Chain system and a supplier by using a RosettaNet system or for integrations between a PeopleSoft eProcurement system and a Commerce One Marketsite by using an XML Common Business Library (XCBL) or SAdt eXplain (SAX) system.

SDK Contents

The PeopleSoft Integration Broker Connector SDK includes:

- Instructions for setting up the development environment.
- Java classes that are required for creating connectors (including IBResponse and IBRequest objects).
- API documentation.
- Sample code for listening and target connector classes.
- A Send Master utility to test connectors.
- A Simple Post utility that enables third-party systems to post messages to the integration gateway.

SDK Location

The following table lists the location of the SDK and its contents.

Item	Location
SDK	<PS_HOME>\webserv\<DOMAIN>\applications\peoplesoft\PSIGW\SDK
Instructions for setting up the connector development environment	<PS_HOME>\webserv\<DOMAIN>\applications\peoplesoft\PSIGW\SDK\docs\ReadMe.txt
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.

Developing Connectors

This section provides overviews of connector development and implementation and general connector class development considerations and discusses how to:

- Develop target connector classes.
- Develop listening connector classes.
- Install connector classes.
- Register connectors.
- Use connector templates.

Understanding Connector Development and Implementation

You can produce new connectors in different ways, based on whether you want to create a listening connector or a target connector.

Listening connectors use standard connector interface and gateway services to link to the integration gateway. Although a Java interface object is not used for listening connectors, the listening connectors still must adhere to a standard scheme of logic to drive requests to, and to process responses from, the integration gateway.

Target connectors must implement a Java interface to become valid target connectors in the integration gateway. This ensures a standard interface for the gateway manager so that it can manage each target connector in a streamlined way.

To develop connectors, you:

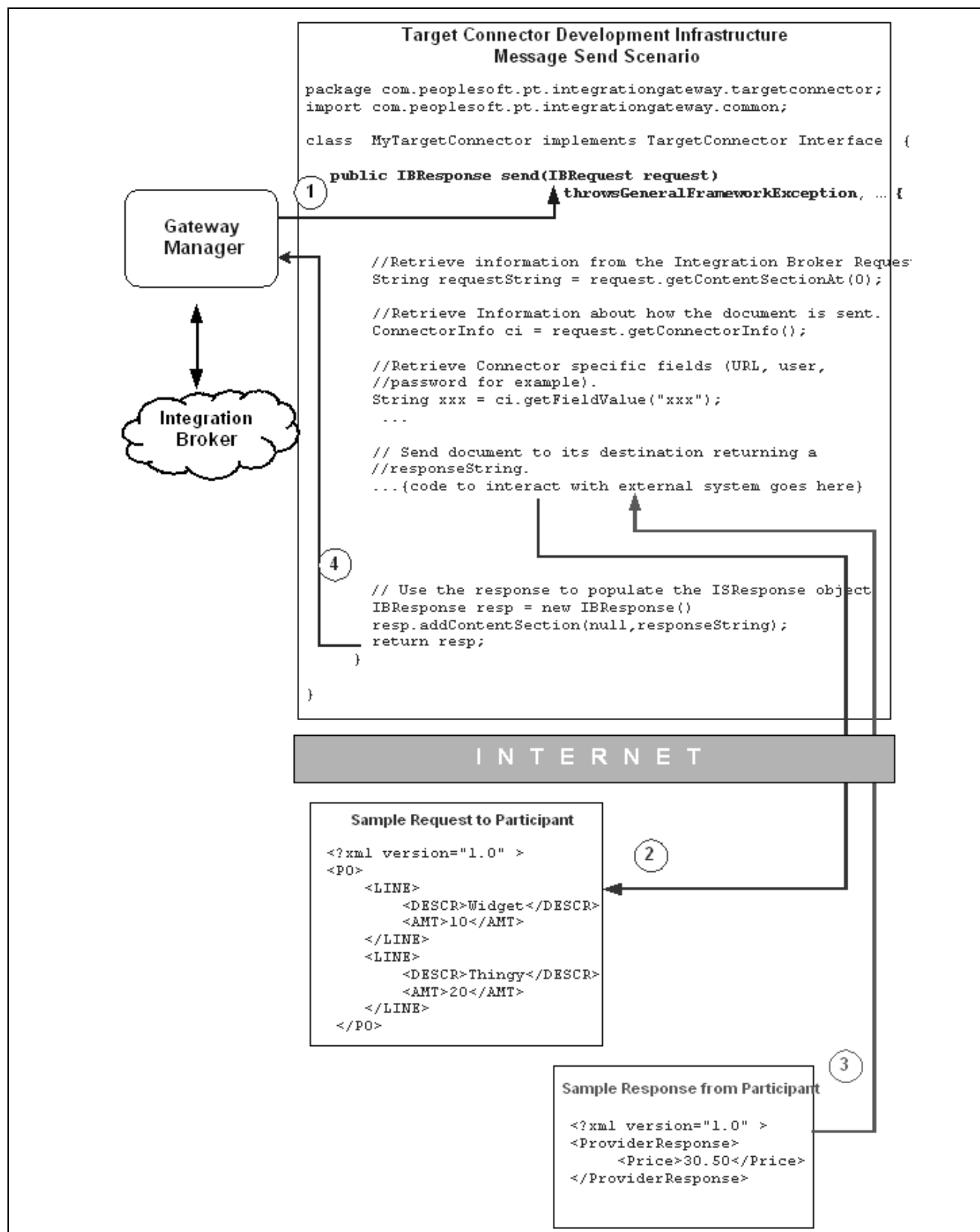
1. Develop a connector class.
2. Install the connector class.
3. Register the connector (target connectors only).

Target and listening connector templates are provided at the end of this chapter for you to use as a starting point for connector development.

Following are various scenarios for the target connector development infrastructure.

Message Send Scenario for the Target Connector Development Infrastructure

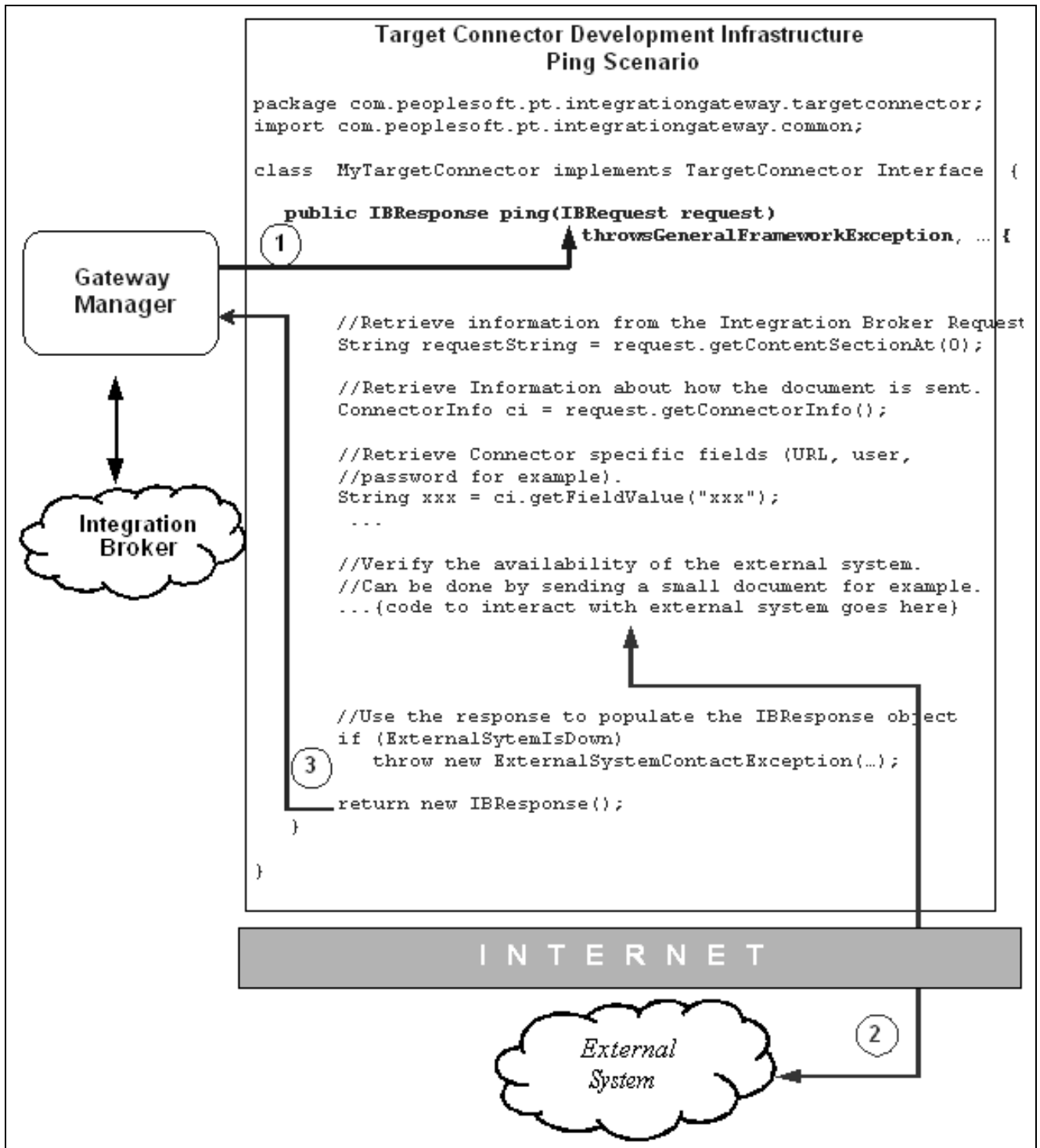
You develop target connectors to generate and send message requests to integration participants and return responses. This diagram shows how the connector code accomplishes these tasks:



Message send scenario

Ping Scenario for the Target Connector Development Infrastructure

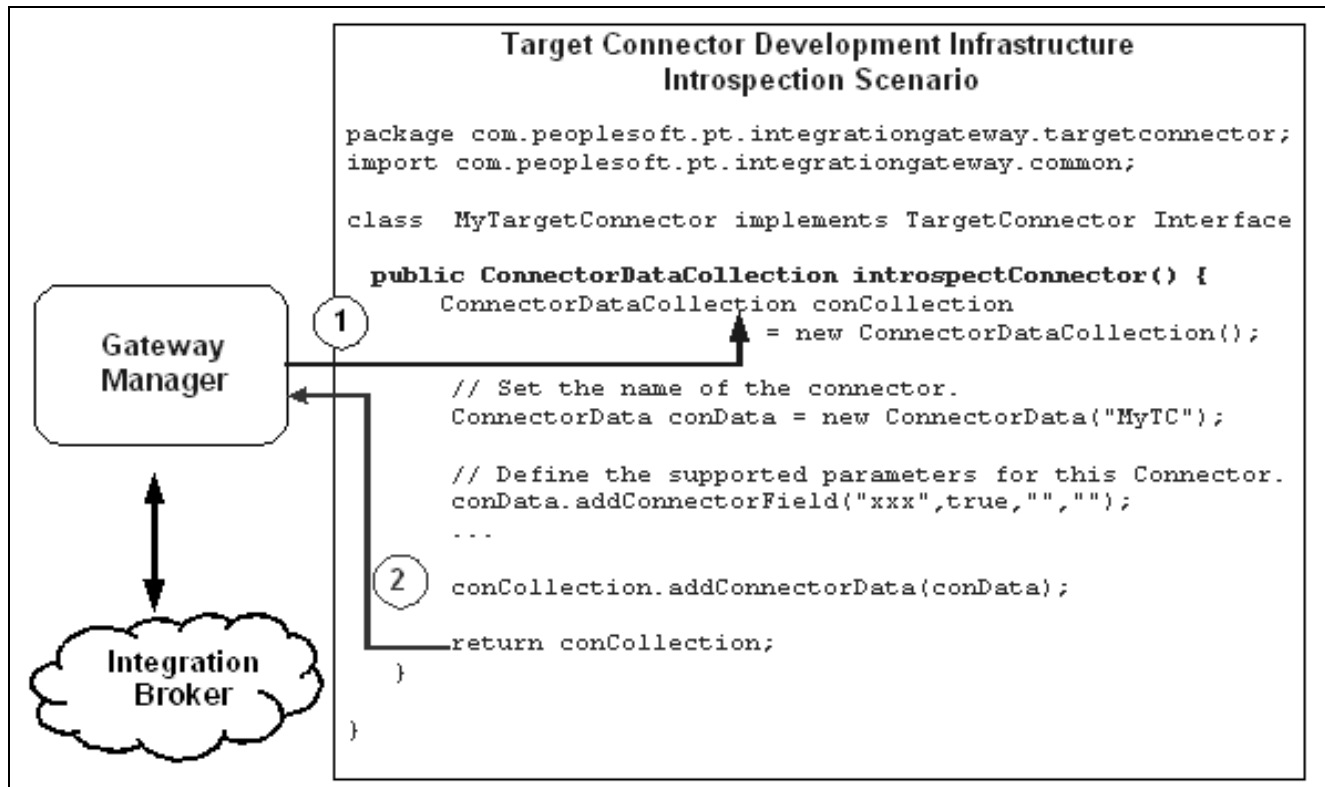
You can develop functionality in target connectors to ping external systems. This diagram shows how the connector code accomplishes this task:



Ping scenario

Introspection Scenario for the Target Connector Development Infrastructure

You can develop functionality within target connectors to handle connector introspection. This diagram shows how the connector code accomplishes this task:



Introspection scenario

Understanding General Connector Class Development Considerations

While implementations vary greatly, when you develop connector classes, you should incorporate specific functionality.

Input and Output Formats That Are Exchanged Through Connectors

For a target connector to handle input and output formats that are exchanged with its intended recipient, it must transform the PeopleSoft Integration Broker request (IBRequest) into a message that is formatted for the intended external system.

For instance, the HTTP target connector that is delivered with PeopleSoft Integration Broker gathers HTTP headers and cookies from the IBRequest and formulates the appropriate HTTP message, complete with the actual message content, so that it can be delivered to its destination. When the response comes back, the connector creates a PeopleSoft Integration Broker response (IBResponse) by using the response string.

For a listening connector to handle input and output formats that are exchanged with its requestor, it must transform the incoming message into an IBRequest. For example, the HTTP listening connector that is delivered with PeopleSoft Integration Broker recognizes SOAP messages and retrieves query string arguments, HTTP headers, and cookies. It then formats all of this information to create the IBRequest so that PeopleSoft Integration Broker can converse with it. When the response comes back, the HTTP listening connector reads the IBResponse and sends its output message content back to the requesting system.

Interaction Between Local and External Systems

A target connector interacts with an external system by sending it information and by retrieving the response.

For example, to accomplish this interaction, the HTTP target connector that is delivered with PeopleSoft Integration Broker uses various HTTP-specific classes to send messages through HTTP and to handle the external system being down, security (through HTTPS), and so forth.

A listening connector interacts with an external system by receiving requests from the external system and returning responses that the external system understands. For example, to accomplish this interaction, the HTTP listening connector that is delivered with PeopleSoft Integration Broker uses a servlet to receive and reply to incoming HTTP messages.

Developing Target Connector Classes

This section discusses target connector class development issues to consider when creating custom target connectors.

Using the Target Connector Interface

As with PeopleSoft-provided target connectors, the integration gateway dynamically invokes custom target connectors through the gateway manager.

Target connectors must adhere to a standard structure as defined in the target connector interface.

```
public interface TargetConnector {
    IBResponse send(IBRequest request) throws
        GeneralFrameworkException,
        DuplicateMessageException,
        InvalidMessageException,
        ExternalSystemContactException,
        ExternalApplicationException,
        MessageMarshallingException,
        MessageUnmarshallingException;

    IBResponse ping(IBRequest request) throws
        GeneralFrameworkException,
        DuplicateMessageException,
        InvalidMessageException,
        ExternalSystemContactException,
        ExternalApplicationException,
        MessageMarshallingException,
        MessageUnmarshallingException;

    ConnectorDataCollection introspectConnector();
}
```

Use the Send method to send a request to an external system and to retrieve its response. The gateway manager passes the request to this method and expects a response to be returned.

The Ping method enables PeopleSoft Integration Broker to verify the availability of a site. The Integration Broker Monitor can also invoke the Ping method explicitly.

ConnectorDataCollection invokes introspection.

Use gateway services, such as XMLDoc, for document access and mutation. You also have access to the IBResponse and IBRequest objects from the integration gateway.

Building Introspection into Target Connectors

PeopleSoft Integration Broker can introspect (query) the capabilities of target connectors that are installed on a local or remote integration gateway by using introspection. Load all target connectors that are delivered with PeopleSoft Integration Broker by clicking the Load button on the Connectors page in the Gateways component.

You can build introspection into custom-built connectors. When you do so, you can load the connector and its properties with the click of a button.

For the introspection process to gather information about a custom target connector, you must implement the IntrospectConnector method.

The following example from the SMTP target connector sends messages through email:

```
public ConnectorDataCollection introspectConnector() {
    //Creates the ConnectorDataCollection that will be returned
    //by this method. This object will contain all the
    //necessary information about this Connector's properties.

    ConnectorDataCollection conCollection = new ConnectorDataCollection();

    //Create ConnectorData Object and stipulating the name of
    //the connector as seen from the Gateway Component.

    ConnectorData conData = new ConnectorData("SMTPTARGET");

    conData.addConnectorField("DestEmailAddress", true, "", "");
    conData.addConnectorField("SourceEmailAddress", true, "", "");
    conData.addConnectorField("CC", false, "", "");
    conData.addConnectorField("BCC", false, "", "");
    conData.addConnectorField("HEADER", "Content-type", false,
        "", "text/plain|text/html");
    conData.addConnectorField("HEADER", "sendUncompressed", true,
        "Y", "Y|N");

    //Add the ConnectorData to your ConnectorDataCollection
    //Object. Typically, you would only
    //add one ConnectorData into your ConnectorDataCollection.
    conCollection.addConnectorData(conData);

    return conCollection;
}
```

Use the addConnectorField method to add connector fields:

```
addConnectorField ([PropertyID] PropertyName, Required, DefaultValue, Possible⇒
Values)
```

Use the following parameters for this method:

Parameter	Description
Property ID	Identifies different property types, such as HEADER for HTTP or SMTP. PeopleSoft software also uses the HEADER property ID to allow a message to be sent in either compressed or clear format through the sendUncompressed property. If this parameter is not supplied, the property ID is the connector name.
Property Name	Identifies the name of the connector property.
Required	Determines whether the information is required for the target connector to deliver its message. Values are: • Y: True • N: False
Default Value	Identifies the default value for the property. Typically, you set the Required parameter to True when you specify a default value so that this information carries to the node configuration in the integration engine.
Possible Values	Identifies a list of the possible values that the property can take, separated by the character.

The following definition shows how these properties function:

```
conData.addConnectorField("HEADER", "sendUncompressed", true, "Y, "Y|N");
```

In this case, the property name is sendUncompressed and its property ID is HEADER. The sendUncompressed property is required (the third parameter is set to true), so that whenever you create a node in the node definition component and specify SMTPTARGET as the target connector, this property appears on the page automatically. Further, because the default value is set to Y, this is the default value. Possible values have been identified as Y or N. If you click the list box (search box) for this property on the Connectors tab in the Node Definition component, you can select from those two values.

Building Error Handling and Logging into Target Connectors

The following code example demonstrates how to build error handling and logging into target connectors:

```
package com.peoplesoft.pt.integrationgateway.targetconnector;

import ...

public class SampleTargetConnector implements TargetConnector {
    public IBResponse ping(IBRequest request)

    public IBResponse send(IBRequest request) throws
        GeneralFrameworkException,
        InvalidMessageException,
        ExternalSystemContactException,
        ExternalApplicationException,
        MessageMarshallingException,
```

```

        MessageUnmarshallingException,
        DuplicateMessageException {

    PSHttp httpObj = null;
    try {

        // Get handle on rootnode
        XmlNode root = dom.GetDocumentElement();

        // Cast the IBRequest back to an InternalIBRequest
        InternalIBRequest request = (InternalIBRequest)requestOrig;

        // Populate App Msg XML Dom Object from IBRequest

        ...

        // Get the URL from either the IBRequest or from the
        //prop file (default)

        String URL = request.getConnectorInfo().getFieldValue("URL");

        // Log the request
        Logger.logMessage("SampleTargetConnector:
        Application Message Request", dom.GenerateXmlString(),
        Logger.STANDARD_INFORMATION);

        // Send the request DOM Document
        httpObj.setRequestProperty("content-type", "text/plain");
        httpObj.send(dom.GenerateXmlString().getBytes());

        // Get the response and convert to a String
        responseString = new String(httpObj.getContent());

        // Log the response
        Logger.logMessage("SampleTargetConnector:
        Application Message Response", responseString,
        Logger.STANDARD_INFORMATION);

        // Construct the IBResponse
        response = new IBResponse();

        ...

        // Return the successful IBResponse
        return response;

    } catch (XmlException xe) {
        httpObj.disconnect();
        throw new GeneralFrameworkException ("SampleTargetConnector:Failed
        while parsing XML");
    }
}

```

```

    } catch (org.w3c.www.protocol.http.HttpException httpException) {
        throw new ("SampleTargetConnector:HTTP Protocol
            exception", httpException);

    } 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 538](#).

This PeopleBook does not discuss how to install servlets on web servers.

See The servlet documentation for your web server.

Invoking Listening Connectors

Listening connectors must invoke PeopleSoft Integration Broker through the gateway manager Connect method.

```

IBResponse connect(IBRequest) throws
    GeneralFrameworkException
    DuplicateMessageException
    InvalidMessageException
    MessageMarshallingException
    MessageUnmarshallingException
    ExternalSystemContactException
    ExternalApplicationException

```

Controlling Message Routing

By accessing and modifying key information on the IBRequest, you can control the behavior of transactions as they flow through the integration gateway.

This section describes several dispatching features that you can use to control message routing by modifying the IBRequest from the listening connector, including routing messages to:

- Other (remote) integration gateways.
- Specific target connectors.
- Other PeopleSoft systems.

You can control the routing of a message to another integration gateway by specifying the uniform resource locator (URL) of the gateway in the `IBRequest`. You might need to forward messages to another gateway so that they can be processed by a remote PeopleSoft Integration Broker system. To do so, specify the URL of this integration gateway as follows:

```
. . .
IBRequest ibRequest = new IBRequest();
IBRequest.setMessageName("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=8.40

#
## JOLT connect string settings for Application Server(s) with known NODENAMES.
#
ig.isc.FINANCIALS.serverURL=//FINSERVER:9000
ig.isc.FINANCIALS.userid=FINUSERID
ig.isc.FINANCIALS.password=FINPASSWORD
ig.isc.FINANCIALS.toolsRel=8.40
```



```
ig.isc.CRM.serverURL=//CRMSEVER:9000
ig.isc.CRM.userid=CRMUSERID
ig.isc.CRM.password=CRMPASSWORD
ig.isc.CRM.toolsRel=8.40
```

Make the following modifications to the listening connector:

```
. . .
IBRequest ibRequest = new IBRequest();
IbRequest.setMessageName("RoutingTest");
IbRequest.setRequestingNode("SourceSystem");
IbRequest.setPassword("myPassword");
. . .

// Get the name of the DestinationSystem from your proprietary
//method. This method would return one of the following
//("HR", //"FINANCIALS", "CRM").
String destinationSystem = getDestinationSystem();
ibRequest.setDestinationNode(destinationSystem);
. . .
// Create a GatewayManager instance.
GatewayManager gm = new GatewayManager();

// Invoke the Connector Framework Manager.
ibResponse = gm.connect(ibRequest);
. . .
```

Building Error Handling and Logging into Listening Connectors

This is sample code for building error handling and logging into listening connectors:

```
package com.peoplesoft.pt.integrationgateway.listeningconnector;

import ...

public class HttpListeningConnector extends HttpServlet {
    public void doGet(HttpServletRequest req,
        HttpServletResponse resp) throws ServletException, IOException {
    }

    public void doPost(HttpServletRequest req,
        HttpServletResponse resp) throws ServletException, IOException {

        String actualResponse = "";
        IBRequest request = null;
        IBResponse response = null;

        try {

            String inputString = MiscUtil.readerToString(new
                InputStreamReader(req.getInputStream()));
```

```
// Log the actual Input String
Logger.logMessage("HttpListeningConnector: HTTP
Request", inputString, Logger.STANDARD_INFORMATION);

HttpListeningConnectorUtility util = new
HttpListeningConnectorUtility();
request = util.createIBRequest("XML", req, inputString);

// Use the GatewayManager to invoke the Integration
// Server and return its response.
GatewayManager conMgr = new GatewayManager();
response = conMgr.connect(request);

// Need to get the actual response from the
//IBResponse
actualResponse = response.getContentSectionAt(0);

} catch (InvalidMessageException ime) {
    ime.printStackTrace();
    actualResponse = getErrorXml(ime);
    Logger.logError("HTTPListeningConnector:
InvalidMessageException", request, response,
Logger.STANDARD_GATEWAY_EXCEPTION, ime);

} catch (ExternalSystemContactException esce) {
    esce.printStackTrace();
    actualResponse = getErrorXml(esce);
    Logger.logError("HTTPListeningConnector:
ExternalSystemContactException", request, response,
Logger.STANDARD_GATEWAY_EXCEPTION, esce);

} catch (ExternalApplicationException esee) {
    esee.printStackTrace();
    actualResponse = getErrorXml(esee);
    Logger.logError("HTTPListeningConnector:
ExternalApplicationException", request, response,
Logger.STANDARD_GATEWAY_EXCEPTION, esee);

} catch (MessageMarshallingException mme) {
    mme.printStackTrace();
    actualResponse = getErrorXml(mme);
    Logger.logError("HTTPListeningConnector:
MessageMarshallingException", request, response,
Logger.STANDARD_GATEWAY_EXCEPTION, mme);

} catch (MessageUnmarshallingException mue) {
    mue.printStackTrace();
    actualResponse = getErrorXml(mue);
    Logger.logError("HTTPListeningConnector:
```

```

        MessageUnmarshallingException", request, response,
        Logger.STANDARD_GATEWAY_EXCEPTION, mue);

    } catch (GeneralFrameworkException gfe) {
        gfe.printStackTrace();
        actualResponse = getErrorXml(gfe);
        Logger.logError("HTTPListeningConnector:
        GeneralFrameworkException", request, response,
        Logger.STANDARD_GATEWAY_EXCEPTION, gfe);
    }

    // Return the message to the original requestor that
    //invoked the Servlet
    HttpListeningConnectorUtility.
        sendResponseBackToRequestor(actualResponse, resp);

    // Log the actual output String
    Logger.logMessage("HttpListeningConnector:
        HTTP Response", actualResponse, Logger.STANDARD_INFORMATION);

    } // end doPost()
}

```

Installing Connector Classes

Install connector classes on the local web server.

Target Connector Classes

To install a target connector class, copy the class from the Java Classes directory to the following location on the local web server:

```
<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.

See Also

[Chapter 4, "Managing Integration Gateways," Registering Installed Target Connectors, page 38](#)

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 IBResponse object
        IBResponse resp = new IBResponse()
        resp.addContentSection(null,responseString);
        return resp;
    }

    public IBResponse ping(IBRequest request) throws
        GeneralFrameworkException
```

```

        DuplicateMessageException
        InvalidMessageException
        ExternalSystemContactException
        ExternalApplicationException
        MessageMarshallingException
        MessageUnmarshallingException    {

    //Retrieve Information about how the document is sent.
    ConnectorInfo ci = request.getConnectorInfo();

    //Retrieve Connector specific fields (URL, user, password
    //for example).
    String xxx = ci.getFieldValue("xxx");
    ...

    // Send a simple document to its destination just to see if
    //the destination is up.
    ...{code to interact with external system goes here. Throw
        exceptions as needed.}

    // Return an empty IBResponse object
    return new IBResponse();
}

public ConnectorDataCollection introspectConnector() {
    ConnectorDataCollection conCollection = new
    ConnectorDataCollection();

    // Set the name of the connector.
    ConnectorData conData = new ConnectorData("MyTC");

    // Define the supported parameters for this Connector.
    conData.addConnectorField("xxx",true,"","");
    ...

    conCollection.addConnectorData(conData);

    return conCollection;
}
}

```

Listening Connector Template (Servlet)

Use the following example as a template for a servlet-based listening connector:

```

package com.peoplesoft.pt.integrationgateway.listeningconnector;
import com.peoplesoft.pt.integrationgateway.common;

//This is an example of using a Servlet when receiving the //External Request.
public class MyListeningConnector extends HttpServlet {

```

```

public void service (HttpServletRequest request,
                    HttpServletResponse response)
                    throws ServletException, IOException {

    //Developer retrieves the request and gets key information
    //here (such as UserName, password, messageName 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.setMessageName(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(MarshallingException me) {

        // Handle Marshalling errors here. For example :
        out.println("<?xml version=\"1.0\"?>"+
            "<MyResponse>"+
            "<Status>"+
            "<Code>1001</Code>"+
            "<Description>"+
            "MarshallingException Occurred"+
            "</Description>"+
            "</Status>"+
            "</MyResponse>");
    } catch(UnmarshallingException ume) {
        // Handle UnmarshallingException here.
    } catch(. . . ) {
        // Handle all other Integration Broker Exceptions.
    } finally {
        //Cleanup work here. For example:
        out.close();
    }
}

```

Listening Connector Template (Non-servlet)

Use the following example as a template for non-servlet-based listening connector:

```
package com.peoplesoft.pt.integrationgateway.listeningconnector;
import com.peoplesoft.pt.integrationgateway.common;

//This is an example of a regular class used to receive the //External Request.
public class MyListeningConnector {

    //Somehow the external system makes a method call to this
    //method. Of course the Class and Method Name is entirely up
    //to the developer.

    public String callMyListeningConnector (String Request) {

        //Developer retrieves the request and gets key information
        //here (such as UserName, password, messageName 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.setMessageName(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(MarshallingException me) {

            // Handle Marshalling errors here. For example :
            return ("<?xml version=\"1.0\"?>"+
                "<MyResponse>"+
                    "<Status>"+
                        "<Code>1001</Code>"+
                        "<Description>"+
                            "MarshallingException Occurred"+
                        "</Description>"+
                    "</Status>"+
```

```

        "</MyResponse>");
    } catch(UnmarshallingException ume) {

        // Handle UnmarshallingException here.
    } catch(. . . ) {

        // Handle all other Integration Broker Exceptions.
    } finally {
        //Cleanup work here (if any). . .
    }
}

```

Developing Connectors Based on DOM

PeopleSoft provides a Java XML DOM wrapper that enables you to manipulate message objects with PeopleCode instead of with a standard XML parser.

The section provides an overview of the Java XML DOM wrapper and discusses how to use Java XML DOM wrapper classes.

Understanding the Java XML DOM Wrapper

The Java XML DOM wrapper is an abstraction layer over XML parsers that enables you to interpret, create, or modify XML messages before you send them into or out of the integration gateway. The XML DOM wrapper provides an API that is equivalent to the PeopleCode API to support composing and transforming XML documents.

For example, suppose that you need to send incoming XML messages to PeopleSoft Integration Broker over HTTP and that you need to read the content of the messages as they come in. Rather than parse each message on a character-by-character basis or use a parser, you can use the Java XML DOM wrapper to read the XML messages as they come into the integration gateway, add information to them as necessary, and pass the messages on to the integration engine.

You can also use the Java XML DOM wrapper to refine XML messages before they are sent, for example, by changing tag names, verifying information in non-XML format, and so forth.

Use the Java XML DOM wrapper to:

- Navigate through XML documents by using methods, such as `GetParentNode`, `GetChild`, and `getSibling`.
- Control the order of elements within XML documents.
- Support additional XML features, such as namespaces and processing instructions.

Using Java XML DOM Wrapper Classes

The following table lists the Java XML DOM wrapper classes. The objects in these classes provide the basic methods for accessing and modifying DOM objects.

Class	Description
XmlDocument	Contains the DOM that supports serialization and deserialization. Provides equivalent methods for dealing with XML documents as PeopleCode does on the application server side.
XmlNode	Provides equivalent methods for dealing with XML nodes as PeopleCode does on the application server side.
XmlNodeList	Provides equivalent methods for dealing with XmlNodeList as PeopleCode does on the application server side.
XmlException	Reports errors that occur while handling the Java XML DOM wrapper classes. These are thrown when an XML DOMException is caught in DOM level. Most of XmlDocument/XmlNode methods throw the XmlException.

Java XML DOM Code Example

The following is a Java XML DOM code example.

```

public String simulateSendingMessage(String message) throws
    GeneralFrameworkException,
    DuplicateMessageException,
    InvalidMessageException,
    ExternalSystemContactException,
    ExternalApplicationException,
    MessageMarshallingException,
    MessageUnmarshallingException,
    XmlException {

    //Create an XmlDocument object.

    //Instantiate a XmlDocument object first. This step is mandatory.
    XmlDocument xmlDoc = new XmlDocument();
    XmlNode rootNode = null;

    //Parse the string into the XmlDocument object.
    try {

        //Fill in the XML structure/data with the real XML string
        xmlDoc.ParseXmlFromString(message);

        //Get the root XmlNode
        rootNode = xmlDoc.GetDocumentElement();
    }
}

```

```

    } catch (XmlException xe) {
        throw new InvalidMessageException
            ("ExampleTargetConnector:InvalidMessageException",xe);
    }

    //Gather credentials from the Xml Document.
    XmlNode itemNode;
    XmlNode tmpNode;
    float price;
    float totalPrice = 0;

    //Read all Message Items and calculate the totalPrice.
    //Get the count of root XmlNode's immediate child XmlNode
    for(int i=0; i < rootNode.GetChildNodeCount(); i++) {

        //Get the number i child XmlNode
        itemNode = rootNode.GetChildNode(i);

        // Only process the Item nodes (any other tag will not be //processed).

        //Get current XmlNode name
        if (!itemNode.GetNodeName().equals("Item")) {
            continue;
        }

        if (itemNode == null) {
            String[] parms = {"Item"};
            throw new InvalidMessageException("ExampleTargetConnector:
                InvalidMessageException", new MessageCatalogEntry(10503,parms),null);
        }

        tmpNode = itemNode.FindNode("Price");

        price = Float.parseFloat(tmpNode.GetNodeValue());

        totalPrice += price * quantity;
    }

    return "<?xml version=\"1.0\"?>"+
        "<ExampleResponse>"+
        "<Status>"+
            "<Code>0</Code>"+
        "</Status>"+
        "<ResponseBody>"+
            "<TotalPrice>"+totalPrice+"</TotalPrice>"+
        "</ResponseBody>"+
        "</ExampleResponse>";
}

```

See Also

Enterprise PeopleTools 8.45 PeopleBook: PeopleCode API Reference, “XmlDoc Class”

Developing and Implementing Connectors in the C/C++ Environment

You should use Java for connector development. However, the PeopleSoft system also provides an environment for developing connectors using C/C++ through the Java Native Interface (JNI).

This section provides an overview of the development process and discusses how to create target connectors for the C/C++ environment.

Understanding the Development Process

To develop connectors for the C/C++ environment, you typically use Xalan and Xerces by Apache, Inc.

Xalan is an XSLT processor that transforms XML documents into HTML, text, or other XML document types. Xalan uses two jar files: xalan.jar and xerces.jar. Xerces is a Java parser to parse XML.

See <http://www.apache.org>.

To develop connectors and not use Xalan and Xerces, you can call a PeopleSoft-delivered Java connector, copy what you need from the message, and pass the information to the C/C++ environment.

To develop connectors for C/C++ environments:

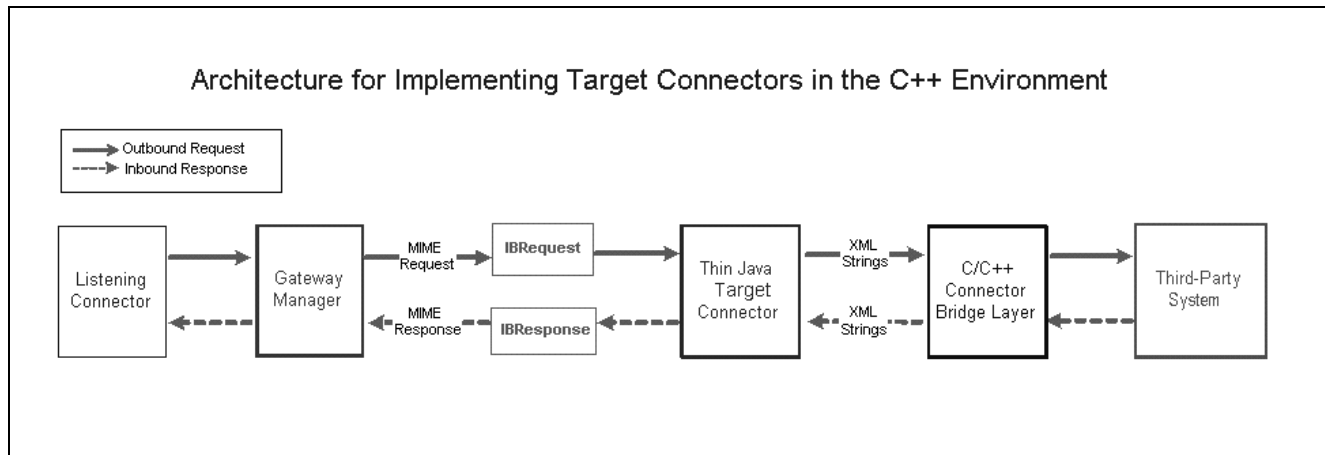
1. Create a Java target connector class.
2. Create a JNI header file.
3. Implement JNI header functions.
4. Build a dynamic-link library (DLL) for the native library.
5. Register the DLL.

Java Target Connectors

Building and implementing a connector in the C/C++ environment includes building a thin Java target connector to enable the gateway manager to load the connector the same way that it loads Java connectors, by parsing the IBRequest object.

The thin Java connector is also a gateway to the C/C++ connector. It is responsible for loading the connector and passing the XML string with business logic to one or more native C/C++ methods.

This diagram illustrates the architecture for implementing target connectors in the C/C++ environment for communication with the PeopleSoft Integration Broker gateway:



Architecture for implementing target connectors in the C/C++ environment

The XML string contains the body from the MIME request string.

The IBResponse object is completely transparent to the native connector. All of the information that the native side needs goes through the XML string or by strings that come from the IBRequest.

PeopleSoft provides a template that you can use as a starting point for developing the connector. In most cases you do not need to make any additions to the code because the IBRequest provides the IBInfo and content body XML strings. However, you can modify the IBInfo and content body XML strings in the C/C++ connector library and declare additional native methods as necessary in the psnativeconnector section of the template.

JNI Headers

After you create a Java connector, you must create a JNI header file by using the javah command. The javah command creates a C/C++ declaration. The JNI header file serves as a bridge between native methods that you call within the Java target connector and those in the C/C++ environment.

JNI Header Functions

When you implement the JNI Header functions, you pass the IBInfo by using a native C/C++ functional call from the Java environment to the third-party system. In doing so, you pass the business logic to the C/C++ system.

DLLs for the Native Library

To build a DLL for the native library, compile the C/C++ functions that are generated by the previous steps.

DLL Registration

Register the DLL that you built for the native library by adding it to the system variables or by adding it to the web server path.

Creating Target Connectors for the C/C++ Environment

To create a target connector for the C/C++ environment:

1. Create a Java target connector.
 - a. Copy the code from the following Java target connector template into a text editor:

```

public class CPlusPlusTargetConnector implements TargetConnector {
    // Native method Declaration
    public native String native_simple(String IBReqString,

```

```

String[] xmlStringArray);

public IBResponse send(IBRequest request) {

    IBResponse response = null;

    try {
        String ibReqString = request.getIBInfoString();
        String requestXml1 = request.getContentSectionAt(0);
        String requestXml2 = request.getContentSectionAt(1);
        . . .

        // Assign to String[] xmlStringArray
        // Load native lib that implements the connector
        System.loadLibrary("psnativeconnector");

        responseStr = native_simple(ibReqString,
            xmlStringArray);

        response = new IBResponse();
        response.addContentSection(null, responseStr);
    }
    catch (Exception ioe) {
        throw new GeneralFrameworkException(ioe.getMessage());
    }
    return response;
}

```

b. Compile the code.

By using a Java compiler, compile the CPlusPlusTargetConnector.java file into a CPlusPlusTargetConnector.class file. Then copy the class file into the target connector directory on the web server. The CLASSPATH environment variable should point to the Integration Broker.jar file.

c. Install the connector.

d. Register the connector.

2. Create a JNI header file.

At a DOS prompt or UNIX shell command prompt, enter the following command and press ENTER:

```

hisdir>javah -jni com.peoplesoft.peoplesoft.pt.integrationgateway.
targetconnector.CplusplusTargetConnector

```

3. Implement the JNI header file.

a. Copy the method declaration output from the previous step to a C++ file.

The following code shows sample method declaration output from the javah command:

```

JNIEXPORT jstring JNICALL Java_com_peoplesoft_pt_
integrationgateway_targetconnector_CplusplusTargetConnector_
native_1simple (JNIEnv *env, jobject obj, jstring ibInfo,
jobjectArray contentArr):

```

- b. Convert the jstring to an ANSI string or to a Unicode string.

To convert the jstring to ANSI, 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 537](#)

[Appendix D, “Using the Integration Broker Connector SDK,” Registering Connectors, page 537](#)

[Chapter 4, “Managing Integration Gateways,” Registering Installed Target Connectors, page 38](#)

Reusing Connector Code

This section discusses how to:

- Reuse connector code through inheritance.
- Reuse connector code through delegation.

Reusing Connector Code Through Inheritance

Use inheritance to extend an existing connector and provide additional behavior.

```
public Class MyHttpTargetConnector extends HttpTargetConnector {
    public IBResponse send(IBRequest request) throws
        GeneralFrameworkException,
        InvalidMessageException,
        ExternalSystemContactException,
        ExternalApplicationException,
        MessageMarshallingException,
        MessageUnmarshallingException,
        DuplicateMessageException {

        // Do connector specific logic here possibly modifying the
        // request . . .
        IBResponse response = super.send(request);

        // Do connector specific logic here possibly modifying the
        // response . . .
    }
}
```

Reusing Connector Code Through Delegation

By using delegation, you can reuse the code from an existing connector to create a new connector, as shown in this example:

```
public Class MyHttpTargetConnector implements TargetConnector {
    public IBResponse ping(IBRequest request) throws
        GeneralFrameworkException,
        InvalidMessageException,
        ExternalSystemContactException,
        ExternalApplicationException,
        MessageMarshallingException,
        MessageUnmarshallingException,
        DuplicateMessageException {

        HttpTargetConnector httpTargetConnector = new
            HttpTargetConnector();
        return httpTargetConnector.ping(request);
    }
    public IBResponse send(IBRequest request) throws
```

```
        GeneralFrameworkException,
        InvalidMessageException,
        ExternalSystemContactException,
        ExternalApplicationException,
        MessageMarshallingException,
        MessageUnmarshallingException,
        DuplicateMessageException {

    // Do connector specific logic here possibly modifying the
    // request . . .
    HttpTargetConnector httpTargetConnector = new
        HttpTargetConnector();
    IBResponse response = httpTargetConnector.send(request);

    // Do connector specific logic here possibly modifying the
    // response . . .
}

public ConnectorDataCollection introspectConnector() {
    HttpTargetConnector httpTargetConnector = new
        HttpTargetConnector();
    ConnectorDataCollection cdc =
        httpTargetConnector.introspectConnector();

    // Possibly add specific fields here . . .
    Return cdc;
}
}
```


APPENDIX E

ISO Country and Currency Codes

PeopleBooks use International Organization for Standardization (ISO) country and currency codes to identify country-specific information and monetary amounts.

This appendix discusses:

- ISO country codes.
- ISO currency codes.

See Also

“About This PeopleBook,” Typographical Conventions and Visual Cues

ISO Country Codes

This table lists the ISO country codes that may appear as country identifiers in PeopleBooks:

ISO Country Code	Country Name
ABW	Aruba
AFG	Afghanistan
AGO	Angola
AIA	Anguilla
ALB	Albania
AND	Andorra
ANT	Netherlands Antilles
ARE	United Arab Emirates
ARG	Argentina
ARM	Armenia
ASM	American Samoa
ATA	Antarctica

ISO Country Code	Country Name
ATF	French Southern Territories
ATG	Antigua and Barbuda
AUS	Australia
AUT	Austria
AZE	Azerbaijan
BDI	Burundi
BEL	Belgium
BEN	Benin
BFA	Burkina Faso
BGD	Bangladesh
BGR	Bulgaria
BHR	Bahrain
BHS	Bahamas
BIH	Bosnia and Herzegovina
BLR	Belarus
BLZ	Belize
BMU	Bermuda
BOL	Bolivia
BRA	Brazil
BRB	Barbados
BRN	Brunei Darussalam
BTN	Bhutan
BVT	Bouvet Island
BWA	Botswana
CAF	Central African Republic
CAN	Canada
CCK	Cocos (Keeling) Islands

ISO Country Code	Country Name
CHE	Switzerland
CHL	Chile
CHN	China
CIV	Cote D'Ivoire
CMR	Cameroon
COD	Congo, The Democratic Republic
COG	Congo
COK	Cook Islands
COL	Colombia
COM	Comoros
CPV	Cape Verde
CRI	Costa Rica
CUB	Cuba
CXR	Christmas Island
CYM	Cayman Islands
CYP	Cyprus
CZE	Czech Republic
DEU	Germany
DJI	Djibouti
DMA	Dominica
DNK	Denmark
DOM	Dominican Republic
DZA	Algeria
ECU	Ecuador
EGY	Egypt
ERI	Eritrea
ESH	Western Sahara

ISO Country Code	Country Name
ESP	Spain
EST	Estonia
ETH	Ethiopia
FIN	Finland
FJI	Fiji
FLK	Falkland Islands (Malvinas)
FRA	France
FRO	Faroe Islands
FSM	Micronesia, Federated States
GAB	Gabon
GBR	United Kingdom
GEO	Georgia
GHA	Ghana
GIB	Gibraltar
GIN	Guinea
GLP	Guadeloupe
GMB	Gambia
GNB	Guinea-Bissau
GNQ	Equatorial Guinea
GRC	Greece
GRD	Grenada
GRL	Greenland
GTM	Guatemala
GUF	French Guiana
GUM	Guam
GUY	Guyana
GXA	GXA - GP Core Country

ISO Country Code	Country Name
GXB	GXB - GP Core Country
GXC	GXC - GP Core Country
GXD	GXD - GP Core Country
HKG	Hong Kong
HMD	Heard and McDonald Islands
HND	Honduras
HRV	Croatia
HTI	Haiti
HUN	Hungary
IDN	Indonesia
IND	India
IOT	British Indian Ocean Territory
IRL	Ireland
IRN	Iran (Islamic Republic Of)
IRQ	Iraq
ISL	Iceland
ISR	Israel
ITA	Italy
JAM	Jamaica
JOR	Jordan
JPN	Japan
KAZ	Kazakstan
KEN	Kenya
KGZ	Kyrgyzstan
KHM	Cambodia
KIR	Kiribati
KNA	Saint Kitts and Nevis

ISO Country Code	Country Name
KOR	Korea, Republic of
KWT	Kuwait
LAO	Lao People's Democratic Rep
LBN	Lebanon
LBR	Liberia
LBY	Libyan Arab Jamahiriya
LCA	Saint Lucia
LIE	Liechtenstein
LKA	Sri Lanka
LSO	Lesotho
LTU	Lithuania
LUX	Luxembourg
LVA	Latvia
MAC	Macao
MAR	Morocco
MCO	Monaco
MDA	Moldova, Republic of
MDG	Madagascar
MDV	Maldives
MEX	Mexico
MHL	Marshall Islands
MKD	Fmr Yugoslav Rep of Macedonia
MLI	Mali
MLT	Malta
MMR	Myanmar
MNG	Mongolia
MNP	Northern Mariana Islands

ISO Country Code	Country Name
MOZ	Mozambique
MRT	Mauritania
MSR	Montserrat
MTQ	Martinique
MUS	Mauritius
MWI	Malawi
MYS	Malaysia
MYT	Mayotte
NAM	Namibia
NCL	New Caledonia
NER	Niger
NFK	Norfolk Island
NGA	Nigeria
NIC	Nicaragua
NIU	Niue
NLD	Netherlands
NOR	Norway
NPL	Nepal
NRU	Nauru
NZL	New Zealand
OMN	Oman
PAK	Pakistan
PAN	Panama
PCN	Pitcairn
PER	Peru
PHL	Philippines
PLW	Palau

ISO Country Code	Country Name
PNG	Papua New Guinea
POL	Poland
PRI	Puerto Rico
PRK	Korea, Democratic People's Rep
PRT	Portugal
PRY	Paraguay
PSE	Palestinian Territory, Occupie
PYF	French Polynesia
QAT	Qatar
REU	Reunion
ROU	Romania
RUS	Russian Federation
RWA	Rwanda
SAU	Saudi Arabia
SDN	Sudan
SEN	Senegal
SGP	Singapore
SGS	Sth Georgia & Sth Sandwich Is
SHN	Saint Helena
SJM	Svalbard and Jan Mayen
SLB	Solomon Islands
SLE	Sierra Leone
SLV	El Salvador
SMR	San Marino
SOM	Somalia
SPM	Saint Pierre and Miquelon
STP	Sao Tome and Principe

ISO Country Code	Country Name
SUR	Suriname
SVK	Slovakia
SVN	Slovenia
SWE	Sweden
SWZ	Swaziland
SYC	Seychelles
SYR	Syrian Arab Republic
TCA	Turks and Caicos Islands
TCD	Chad
TGO	Togo
THA	Thailand
TJK	Tajikistan
TKL	Tokelau
TKM	Turkmenistan
TLS	East Timor
TON	Tonga
TTO	Trinidad and Tobago
TUN	Tunisia
TUR	Turkey
TUV	Tuvalu
TWN	Taiwan, Province of China
TZA	Tanzania, United Republic of
UGA	Uganda
UKR	Ukraine
UMI	US Minor Outlying Islands
URY	Uruguay
USA	United States

ISO Country Code	Country Name
UZB	Uzbekistan
VAT	Holy See (Vatican City State)
VCT	St Vincent and the Grenadines
VEN	Venezuela
VGB	Virgin Islands (British)
VIR	Virgin Islands (U.S.)
VNM	Viet Nam
VUT	Vanuatu
WLF	Wallis and Futuna Islands
WSM	Samoa
YEM	Yemen
YUG	Yugoslavia
ZAF	South Africa
ZMB	Zambia
ZWE	Zimbabwe

ISO Currency Codes

This table lists the ISO country codes that may appear as currency identifiers in PeopleBooks:

ISO Currency Code	Description
ADP	Andorran Peseta
AED	United Arab Emirates Dirham
AFA	Afghani
AFN	Afghani
ALK	Old Lek
ALL	Lek
AMD	Armenian Dram

ISO Currency Code	Description
ANG	Netherlands Antilles Guilder
AOA	Kwanza
AOK	Kwanza
AON	New Kwanza
AOR	Kwanza Reajustado
ARA	Austral
ARP	Peso Argentino
ARS	Argentine Peso
ARY	Peso
ATS	Schilling
AUD	Australian Dollar
AWG	Aruban Guilder
AZM	Azerbaijani Manat
BAD	Dinar
BAM	Convertible Marks
BBD	Barbados Dollar
BDT	Taka
BEC	Convertible Franc
BEF	Belgian Franc
BEL	Financial Belgian Franc
BGJ	Lev A/52
BGK	Lev A/62
BGL	Lev
BGN	Bulgarian LEV
BHD	Bahraini Dinar
BIF	Burundi Franc
BMD	Bermudian Dollar

ISO Currency Code	Description
BND	Brunei Dollar
BOB	Boliviano
BOP	Peso
BOV	Mvdol
BRB	Cruzeiro
BRC	Cruzado
BRE	Cruzeiro
BRL	Brazilian Real
BRN	New Cruzado
BRR	Brazilian Real Dollar
BSD	Bahamian Dollar
BTN	Ngultrum
BUK	N/A
BWP	Pula
BYB	Belarussian Ruble
BYR	Belarussian Ruble
BZD	Belize Dollar
CAD	Canadian Dollar
CDF	Franc Congolais
CHF	Swiss Franc
CLF	Unidades de fomento
CLP	Chilean Peso
CNX	Peoples Bank Dollar
CNY	Yuan Renminbi
COP	Colombian Peso
CRC	Costa Rican Colon
CSD	Serbia Dinar

ISO Currency Code	Description
CSJ	Krona A/53
CSK	Koruna
CUP	Cuban Peso
CVE	Cape Verde Escudo
CYP	Cyprus Pound
CZK	Czech Koruna
DEM	Deutsche Mark
DJF	Djibouti Franc
DKK	Danish Krone
DOP	Dominican Peso
DZD	Algerian Dinar
ECS	Sucre
ECV	Unidad de Valor
EEK	Kroon
EGP	Egyptian Pound
EQE	Ekwele
ERN	Nakfa
ESA	Spanish Peseta
ESB	Convertible Peseta
ESP	Spanish Peseta
ETB	Ethiopian Birr
EUR	euro
FIM	Markka
FJD	Fiji Dollar
FKP	Falklands Isl. Pound
FRF	French Franc
GBP	Pound Sterling

ISO Currency Code	Description
GEK	Georgian Coupon
GEL	Lari
GHC	Cedi
GIP	Gibraltar Pound
GMD	Dalasi
GNE	Syli
GNF	Guinea Franc
GNS	Syli
GQE	Ekwele
GRD	Drachma
GTQ	Quetzal
GWE	Guinea Escudo
GWP	Guinea-Bissau Peso
GYD	Guyana Dollar
HKD	Hong Kong Dollar
HNL	Lempira
HRD	Dinar
HRK	Kuna
HTG	Gourde
HUF	Forint
IDR	Rupiah
IEP	Irish Pound
ILP	Pound
ILR	Old Shekel
ILS	New Israeli Sheqel
INR	Indian Rupee
IQD	Iraqi Dinar

ISO Currency Code	Description
IRR	Iranian Rial
ISJ	Old Krona
ISK	Iceland Krona
ITL	Italian Lira
JMD	Jamaican Dollar
JOD	Jordanian Dinar
JPY	Yen
KES	Kenyan Shilling
KGS	Som
KHR	Riel
KMF	Comoro Franc
KPW	North Korean Won
KRW	Won
KWD	Kuwaiti Dinar
KYD	Cayman Islands dollar
KZT	Tenge
LAJ	Kip Pot Pol
LAK	Kip
LBP	Lebanese Pound
LKR	Sri Lanka Rupee
LRD	Liberian Dollar
LSL	Loti
LSM	Maloti
LTN	Lithuanian Litas
LTT	Talonas
LUC	Convertib Franc
LUF	Luxembourg Franc

ISO Currency Code	Description
LUL	Financial Franc
LVL	Latvian Lats
LVR	Latvian Ruble
LYD	Libyan Dinar
MAD	Moroccan Dirham
MAF	Mali Franc
MDL	Moldovan Leu
MGF	Malagasy Franc
MKD	Denar
MLF	Mali Franc
MMK	Kyat
MNT	Tugrik
MOP	Pataca
MRO	Ouguiya
MTL	Maltese Lira
MTP	Maltese Pound
MUR	Mauritius Rupee
MVQ	Maldiva Rupee
MVR	Rufiyaa
MWK	Malawian Kwacha
MXN	Mexican Peso
MXP	Mexican Peso
MXV	Mexican UDI
MYR	Malaysian Ringgit
MZE	Mozambique Escudo
MZM	Metical
NAD	Namibia Dollar

ISO Currency Code	Description
NGN	Naira
NIC	Cordoba
NIO	Cordoba Oro
NLG	Netherlands Guilder
NOK	Norwegian Krone
NPR	Nepalese Rupee
NZD	New Zealand Dollar
OMR	Rial Omani
PAB	Balboa
PEI	Inti
PEN	Nuevo Sol
PES	Sol
PGK	Kina
PHP	Philippine Peso
PKR	Pakistan Rupee
PLN	Zloty
PLZ	Zloty
PTE	Portuguese Escudo
PYG	Guarani
QAR	Qatari Rial
ROK	Leu A/52
ROL	Leu
RUB	Russian Ruble
RUR	Russian Federation Rouble
RWF	Rwanda Franc
SAR	Saudi Riyal
SBD	Solomon Islands

ISO Currency Code	Description
SCR	Seychelles Rupee
SDD	Sudanese Dinar
SDP	Sudanese Pound
SEK	Swedish Krona
SGD	Singapore Dollar
SHP	St Helena Pound
SIT	Tolar
SKK	Slovak Koruna
SLL	Leone
SOS	Somali Shilling
SRG	Surinam Guilder
STD	Dobra
SUR	Rouble
SVC	El Salvador Colon
SYP	Syrian Pound
SZL	Lilangeni
THB	Baht
TJR	Tajik Ruble
TJS	Somoni
TMM	Manat
TND	Tunisian Dinar
TOP	Pa'anga
TPE	Timor Escudo
TRL	Turkish Lira
TTD	Trinidad Dollar
TWD	New Taiwan Dollar
TZS	Tanzanian Shilling

ISO Currency Code	Description
UAH	Hryvnia
UAK	Karbovanet
UGS	Uganda Shilling
UGW	Old Shilling
UGX	Uganda Shilling
USD	US Dollar
USN	US Dollar (Next day)
USS	US Dollar (Same day)
UYN	Old Uruguay Peso
UYP	Uruguayan Peso
UYU	Peso Uruguayo
UZS	Uzbekistan Sum
VEB	Bolivar
VNC	Old Dong
VND	Dong
VUV	Vatu
WST	Tala
XAF	CFA Franc BEAC
XAG	Silver
XAU	GOLD
XBA	European Composite Unit
XBB	European Monetary Unit
XBC	European Unit of Account 9
XBD	European Unit of Account 17
XCD	East Caribbean Dollar
XDR	SDR
XEU	EU Currency (E.C.U)

ISO Currency Code	Description
XFO	Gold-Franc
XFU	UIC-Franc
XOF	CFA Franc BCEAO
XPD	Palladium
XPF	CFP Franc
XPT	Platinum
XTS	For Testing Purposes
XXX	Non Currency Transaction
YDD	Yemeni Din
YER	Yemeni Rial
YUD	New Yugoslavian Dinar
YUM	New Dinar
YUN	Yugoslavian Dinar
ZAL	Financial Rand
ZAR	Rand
ZMK	Zambian Kwacha
ZRN	New Zaire
ZRZ	Zaire
ZWC	Rhodesian Dollar
ZWD	Zimbabwe Dollar

Glossary of PeopleSoft 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.
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 Human Resources, PeopleSoft Benefits Administration, PeopleSoft Stock Administration, and the COBRA Administration feature of the Base Benefits business process.
action template	In PeopleSoft 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 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.

agreement	In PeopleSoft 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 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.
AR specialist	Abbreviation for <i>receivables specialist</i> . In PeopleSoft Receivables, an individual in who tracks and resolves deductions and disputed items.
arbitration plan	In PeopleSoft Enterprise Pricer, defines how price rules are to be applied to the base price when the transaction is priced.
assessment rule	In PeopleSoft 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 Directory Interface, relates the data that makes up an entry in the directory information tree.
authentication server	A server that is set up to verify users of the system.
base time period	In PeopleSoft Business Planning, the lowest level time period in a calendar.
benchmark job	In PeopleSoft Workforce Analytics, a benchmark job is a job code for which there is corresponding salary survey data from published, third-party sources.
book	In PeopleSoft 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."
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 event	In PeopleSoft Receivables, defines the processing characteristics for the Receivable Update process for a draft activity.

	In PeopleSoft 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 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 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.
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 Catalog Management, translates values from the catalog source data to the format of the company's catalog.
catalog partner	In PeopleSoft 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.
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 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 Promotions Management, enables you to view financial data (such as planned, incurred, and actual amounts) that is related to funds and trade promotions.
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 <i>sub-classification</i> .
clone	In PeopleCode, to make a unique copy. In contrast, to <i>copy</i> may mean making a new reference to an object, so if the underlying object is changed, both the copy and the original change.
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, PeopleSoft maintains a set of collections (one per language code) for each search index object.

collection rule	In PeopleSoft 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.
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.
condition	In PeopleSoft 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. 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.
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 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 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.
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 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.

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 Supply Chain Management, identifies the method by which goods are shipped to their destinations (such as truck, air, rail, and so on). 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.
directory information tree	In PeopleSoft Directory Interface, the representation of a directory's hierarchical structure.
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 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 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 <i>Enterprise Incentive Management ledger</i> . 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.
elimination set	In PeopleSoft General Ledger, a related group of intercompany accounts that is processed during consolidations.
entry event	In PeopleSoft 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 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.
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 Human Resources, also refers to an incident that affects benefits eligibility.
event propagation process	In PeopleSoft 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.

	Sales Incentive Management uses this mechanism to implement splits, roll-ups, and so on. Event propagation determines who receives the credit.
exception	In PeopleSoft Receivables, an item that either is a deduction or is in dispute.
exclusive pricing	In PeopleSoft 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 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.
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.
fund	In PeopleSoft 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.
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.
group	In PeopleSoft Billing and Receivables, a posting entity that comprises one or more transactions (items, deposits, payments, transfers, matches, or write-offs). In PeopleSoft 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 Time and Labor, for example, employees are placed in groups for time reporting purposes.
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, user interaction objects, and so on.
incentive rule	In PeopleSoft 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 Promotions Management, to become liable for a promotional payment. In other words, you owe that amount to a customer for promotional activities.
item	In PeopleSoft Inventory, a tangible commodity that is stored in a business unit (shipped from a warehouse). In PeopleSoft 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.
KPI	In PeopleSoft Receivables, an individual receivable. An item can be an invoice, a credit memo, a debit memo, a write-off, or an adjustment. An abbreviation for <i>key performance indicator</i> . 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.

LDIF file	Abbreviation for <i>Lightweight Directory Access Protocol (LDAP) Data Interchange Format file</i> . Contains discrepancies between PeopleSoft 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.
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 <i>rates</i>) 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.
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.
load	In PeopleSoft Inventory, identifies a group of goods that are shipped together. Load management is a feature of PeopleSoft Inventory that is used to track the weight, the volume, and the destination of a shipment.
local functionality	In PeopleSoft 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 <i>1</i> —is the address you use most often and may be different from the main address.
logistical task	In PeopleSoft 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.
match group	In PeopleSoft 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 <i>PeopleSoft MultiChannel Framework server</i> . Comprises the universal queue server and the MCF log server. Both processes are started when <i>MCF Servers</i> is selected in an application server domain configuration.
merchandising activity	In PeopleSoft 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.
meta-SQL	Meta-SQL constructs expand into platform-specific Structured Query Language (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	Metastings are special expressions included in SQL string literals. The metastings, 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 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 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.
node-oriented tree	A tree that is based on a detail structure, but the detail values are not used.
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 and non-PeopleSoft 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 <i>compensation object</i> .
partner	A company that supplies products or services that are resold or purchased by the enterprise.
pay cycle	In PeopleSoft Payables, a set of rules that define the criteria by which it should select scheduled payments for payment creation.
pending item	In PeopleSoft 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.

PeopleCode	PeopleCode is a proprietary language, executed by the PeopleSoft application processor. PeopleCode generates results based upon existing data or user actions. By using business interlink objects, external services are available to all PeopleSoft applications wherever PeopleCode can be executed.
PeopleCode event	An action that a user takes upon an object, usually a record field, that is referenced within a PeopleSoft page.
PeopleSoft 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.
plan	In PeopleSoft 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 Supply Planning, a set of data (business units, items, supplies, and demands) constituting the inputs and outputs of a supply plan.
portal registry	In PeopleSoft 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.
price list	In PeopleSoft Enterprise Pricer, 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	In PeopleSoft Enterprise Pricer, defines 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 condition	In PeopleSoft Enterprise Pricer, selects the price-by fields, the values for the price-by fields, and the operator that determines how the price-by fields are related to the transaction.
price rule key	In PeopleSoft Enterprise Pricer, defines the fields that are available to define price rule conditions (which are used to match a transaction) on the price rule.
process category	In PeopleSoft Process Scheduler, processes that are grouped for server load balancing and prioritization.
process group	In PeopleSoft 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.
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 category	In PeopleSoft Enterprise Incentive Management, indicates an application in the Enterprise Incentive Management suite of products. Each transaction in the PeopleSoft Enterprise Incentive Management system is associated with a product category.
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 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 Project Costing, an individual transaction line that represents a cost, time, budget, or other transaction row.
promotion	In PeopleSoft Promotions Management, a trade promotion, which is typically funded from trade dollars and used by consumer products manufacturers to increase sales volume.
publishing	In PeopleSoft Enterprise Incentive Management, a stage in processing that makes incentive-related results available to participants.
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 <i>record input value-added tax flag</i> . Within PeopleSoft 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 Order Management, Billing, or Receivables, where it is assumed that you are always recording only output VAT, or in PeopleSoft Expenses, where it is assumed that you are always recording only input VAT.
record output VAT flag	Abbreviation for <i>record output value-added tax flag</i>. See <i>record input VAT flag</i>.
reference data	In PeopleSoft Sales Incentive Management, system objects that represent the sales organization, such as territories, participants, products, customers, channels, and so on.
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 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.
REN server	Abbreviation for <i>real-time event notification server</i> in PeopleSoft MultiChannel Framework.
requester	In PeopleSoft eSettlements, an individual who requests goods or services and whose ID appears on the various procurement pages that reference purchase orders.
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.
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.
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.
serial genealogy	In PeopleSoft Manufacturing, the ability to track the composition of a specific, serial-controlled item.
serial in production	In PeopleSoft Manufacturing, enables the tracing of serial information for manufactured items. This is maintained in the Item Master record.
session	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 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.
single signon	With single signon, users can, after being authenticated by a PeopleSoft application server, access a second PeopleSoft application server without entering a user ID or password.
source transaction	In commitment control, any transaction generated in a PeopleSoft 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.
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.

statutory account	Account required by a regulatory authority for recording and reporting financial results. In PeopleSoft, this is equivalent to the Alternate Account (ALTACCT) ChartField.
step	In PeopleSoft Sales Incentive Management, a collection of sections in a plan. Each step corresponds to a step in the job run.
storage level	In PeopleSoft 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 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.
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 <i>basis</i> 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 Receivables, an activity that defines how the system generates accounting entries for the general ledger.
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.
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, you use templates to build a page by combining HTML from a number of sources. For a PeopleSoft portal, all templates must be registered in the portal registry, and each content reference must be assigned a template.
territory	In PeopleSoft Sales Incentive Management, hierarchical relationships of business objects, including regions, products, customers, industries, and participants.
TimeSpan	A relative period, such as year-to-date or current period, that can be used in various PeopleSoft General Ledger functions and reports when a rolling time frame, rather

	than a specific date, is required. TimeSpans can also be used with flexible formulas in PeopleSoft Projects.
trace usage	In PeopleSoft 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 systems that displays the relationship between all accounting units (for example, corporate divisions, projects, reporting groups, account numbers) and determines roll-up hierarchies.
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 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.
user interaction object	In PeopleSoft Sales Incentive Management, used to define the reporting components and reports that a participant can access in his or her context. All 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 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 <i>value-added tax exception</i> . 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 <i>value-added tax exempt</i> . 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 <i>value-added tax exoneration</i> . An organization that has been granted a permanent exemption from paying VAT due to the nature of that organization.
VAT suspension	Abbreviation for <i>value-added tax suspension</i> . An organization that has been granted a temporary exemption from paying VAT.
warehouse	A PeopleSoft data warehouse that consists of predefined ETL maps, data warehouse tools, and DataMart definitions.

work order	In PeopleSoft 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.
worksheet	A way of presenting data through a PeopleSoft 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 schema	An XML definition that standardizes the representation of application messages, component interfaces, or business interlinks.
yield by operation	In PeopleSoft Manufacturing, the ability to plan the loss of a manufactured item on an operation-by-operation basis.
zero-rated VAT	Abbreviation for <i>zero-rated value-added tax</i> . 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.

Index

A

- adapters, iWay SOAPswitch, *See* ERP connectors
- additional documentation xx
- aliases
 - creating for records 148
 - issuer aliases for root certificates 398
 - preserving for records and fields 254
 - understanding 67
- Apache Xalan 250, 545
- Apache Xerces 545
- Application Designer
 - accessing integration routing events 159
 - activating subscription PeopleCode 244
 - creating message subscriptions 161
 - creating test messages 478
 - creating transform programs 478
 - defining message channels 138
 - defining messages 143
 - defining transform programs 249
- Application Engine transform programs,
See transformations
- application fundamentals xix
- application servers
 - applying transformations 50
 - generating requests 14
 - handling exceptions 358
 - logging 357
 - Publication Broker 293
 - See Also* Publication Broker
 - Publication Contractor 293
 - See Also* Publication Contractor
 - receiving requests 12
 - returning responses 13
 - setting up digital certificates for server authentication 409
 - Subscription Contractor 293
 - See Also* Subscription Contractor
 - tracing 357
- archiving
 - asynchronous messages 339
 - deleting archived messages 348
 - message instances 336
 - messages assigned to a channel 140
 - running batch message archiving processes 347
 - searching for archived messages 306
 - setting error logging properties 50
 - setting system message logging properties 49
- asynchronous
 - integrations 21
 - See Also* asynchronous integrations
 - messaging/transmission 21
 - See Also* asynchronous messaging
 - transactions 10
 - See Also* asynchronous transactions
- asynchronous integrations
 - configuring messaging servers 21
 - defining messages 137
 - understanding 217
 - viewing information 237
- asynchronous messaging
 - brokers, contractors, queues 293
 - converting messages to synchronous 275
 - debugging transformations 358
 - displaying status 301
 - handling cookies 168
 - hub routing 275
 - monitoring instances 305, 308
 - monitoring message details 336
 - See Also* Message Details component
 - partitioning channels 141
 - See Also* partitioning
 - publishing 295
 - refreshing queues 30
 - responding to request messages 98
 - retrieving XML representations 478
 - running batch error notifications 344
 - server processes 294
 - simulating receiving messages from external nodes 185
 - submitting non-SOAP messages 99
 - transforming messages 276
 - understanding compression 65
 - understanding dedicated servers 23
 - understanding subscription 298

- using JMS target connectors 111
- viewing errors 341
- viewing in XML format 341
- viewing inbound post statistics 324
- viewing nonrepudiation information 341
- viewing performance statistics 305, 316, 317
- viewing properties 337
- asynchronous transactions
 - asynchronous-to-synchronous transactions 217
 - See Also* asynchronous-to-synchronous transactions
 - handling inbound 171
 - handling outbound 163
 - Integration Point Wizard 219
 - See Also* Integration Point Wizard
 - iWay SOAPswitch operations 364
 - See Also* iWay SOAPswitch
 - selecting channels for service messages (WSDL) 380
 - selecting messages for services (WSDL) 378
 - setting properties 224
 - specifying receiver message information 222
 - specifying reply message information 226
 - specifying sender message information 219
 - specifying transformation information 226
 - transaction modifiers 274
 - See Also* transaction modifiers
 - transforming messages 220
 - understanding inbound 10
 - understanding integrations 217
 - understanding outbound 10
 - viewing integration points 227
- asynchronous-to-synchronous integrations
 - understanding 217
 - viewing integration information 240
- asynchronous-to-synchronous transactions
 - setting transaction properties 224
 - specifying receiver message information 222
 - specifying sender message information 219

- specifying transformation information 220, 226
- understanding 217
- viewing integration points 230
- audit action codes 71
- authentication
 - authenticating nodes 199
 - configuring 399
 - digital certificates 387
 - See Also* digital certificates
 - implementing 394
 - implementing password authentication 399
 - running HTTP transactions through proxy servers 100
 - setting for FTPS connectors 119
 - understanding 387

B

- Baltimore Technologies 389
- batch operations
 - running error notifications 344
 - running message archiving processes 347
- BEA Jolt, *See* Jolt
- BEA Tuxedo, *See* Tuxedo
- BEA WebLogic, *See* WebLogic
- bindings 362, 363
- broker dispatchers/handlers
 - asynchronous publishing of instances 295
 - asynchronous subscription of instances 298
 - server processes 294
 - understanding 317
 - viewing statistics 325
- Broker Handler page 325
- built-in functions
 - bypassing the integration engine to send messages 54
 - ConnectorRequest 54
 - ConnectorRequestURL 55
 - Exit 188
 - FindCodeSetValues 270, 488
 - GetMessage 171, 184
 - GetMessageXmlDoc 171, 183, 184
 - GetXMLDoc 191, 192
 - PublishXmlDoc 163, 164, 432
 - ReturnToServer 156
 - SetXMLDoc 76, 191

SyncRequestXmlDoc 165, 166, 208
 Transform 498

C

C/C++

building DLLs for native library 546
 creating JNI headers/header
 functions 546
 creating target connectors 546
 developing Java target connectors 545
 registering DLLs for native library 546
 understanding connector
 development 545

CA

accessing certificate properties 391
 adding CA authorities 410
 installing node-based digital
 certificates 394
 installing root certificates 395
 installing signed public key
 certificates 395
 obtaining signed public encryption
 keys 391
 resolving root certificate
 mismatches 398
 understanding 389
 understanding root certificates 391

CDATA

generating outbound messages 163
 handling non-XML data 252
 passing HTTP parameters 94
 sending non-XML files 76
 submitting non-XML messages 93
 understanding 93
 using for transformations 493

certificate authorities (CA), *See* CA

certificate signing request (CSR), *See* CSR

certificates

certificate authorities (CA) 389
 See Also CA
 certificate signing request (CSR) 390
 See Also CSR
 digital 387
 See Also digital certificates
 public key 391
 See Also public key certificates
 remote 398
 See Also remote certificates
 root 391
 See Also root certificates

setting security properties 45
 setting up local node 410
 setting up remote node 411

channels

activating 244
 archiving messages 140
 assigning messages 150
 assigning messages to 150
 assigning messaging servers 23, 25, 26
 Channel Status page 311
 configuring 139
 creating for service messages
 (WSDL) 381
 debugging 358
 defining 138
 deleting during upgrade 143
 editing messaging server channel
 lists 27
 granting access to 142
 handling nonrepudiation messages 97
 partitioning 140
 See Also partitioning
 processing messages in parallel 140
 selecting for service messages
 (WSDL) 380
 selecting messages for services
 (WSDL) 378
 selecting status 139
 sending messages in order 140
 setting subchannels for JMS queue
 listeners 105
 setting subchannels for JMS topic
 subscribers 106
 understanding 15, 138
 understanding blocked 303
 understanding stalled 303
 viewing status 305

classes

developing for connectors 528
 developing for listening connectors 533
 developing for target connectors 529
 installing connector classes 537
 Message 67, 70, 75, 137, 187
 Rowset 67, 75, 93, 137, 254
 SoapDoc 75, 137, 147, 166, 185, 187
 TransformData 251, 252
 using Java XML DOM wrapper
 classes 542
 XmlDoc 70, 75, 137, 187, 255

clean-up mode 313, 314

- client authentication
 - generating private and public key pairs 413
 - installing digital certificates for 412
- Codeset Group page 261
- codeset groups
 - creating 481
 - defining 261
 - selecting for nodes 200, 490
 - understanding 260, 480
- Codeset page 262
- codeset repositories
 - elements 260
 - translating via 259, 479
 - understanding 480
- codeset values
 - creating 482
 - defining 263
 - deleting 265
 - understanding 260, 480
- Codeset Values page 263
- codesets
 - codeset groups 260
 - See Also* codeset groups
 - codeset values 260
 - See Also* codeset values
 - creating 482
 - defining 262
 - deleting 265
 - exporting/importing 264
 - implementing via PeopleCode 488
 - implementing via XSLT 485
 - repository 260
 - See Also* codeset repositories
 - understanding 260, 480
- comments, submitting xxiii
- common elements xxiii
- components
 - Gateways component 36
 - See Also* Gateways component
 - generating WSDL from component interfaces 383, 384
 - iWay SOAPswitch 363
 - Monitor Message component 304
 - See Also* Monitor Message component
- compression
 - applying transformations at gateway 51
 - setting for FTP target connectors 117
 - setting for HTTP connectors 91
 - setting for JMS messages 112
 - setting for PeopleSoft 8.1 listening connectors 101
 - setting for simple file target connectors 116
 - setting for SMTP target connectors 123
 - setting message properties 85
 - setting the message compression threshold 65
- Connector Properties page 39
- connector SDK
 - API documentation 524
 - contents 524
 - developing connectors 525
 - developing connectors based on DOM 542
 - developing connectors in C/C++ 545
 - locating SDK and its contents 524
 - registering connectors 38
 - reusing connector code 549
 - understanding 8, 523
- connectors
 - configuring integration gateways 36
 - connector management service 9
 - developing 525
 - developing based on DOM 542
 - developing classes for 528
 - developing in C/C++ environment 545
 - editing transaction connector properties 212
 - ERP 363
 - See Also* ERP connectors
 - exchanging input/output formats 528
 - incoming request message flow through the architecture 11
 - installing classes 537
 - interacting between local and external systems 529
 - listening 8
 - See Also* listening connectors
 - outgoing request message flow through the architecture 11
 - overrides in transaction modifiers 283
 - overriding transaction-level properties 222, 223, 224
 - receiving third-party messages 442, 446
 - reusing code 549
 - SDK 523
 - See Also* connector SDK
 - selecting for transactions 210

- setting gateway properties 46
- target 8
 - See Also* target connectors
- understanding 7
- understanding delivered 79
- understanding gateway manager 8
 - See Also* gateway manager
- using templates 538
- Connectors page 212
- Consumer System Definition Wizard 369
- contact information xxiii
- contacts
 - overriding for relationships 279
 - specifying for nodes 201
- cookies
 - handling in messages 168
 - receiving/returning 168
 - submitting in HTTP headers 97
- Create Test Message feature 252
- creating integrations overview 16
- cross-references xxii
- CSR
 - generating (BEA WebLogic) 402
 - generating (IBM WebSphere) 406
 - generating for client authentication 414
 - installing signed public key
 - certificates 395
 - submitting to CAs for signing (BEA WebLogic) 404
 - submitting to CAs for signing (IBM WebSphere) 407
 - understanding 390
- Customer Connection website xx

D

- databases
 - importing/exporting codesets
 - between 264
 - local/remote nodes 196
 - See Also* nodes
 - MySQL for iWay SOAPswitch 364
 - understanding record fields 141
- DB2
 - importing WSDL 372
 - setting persistent cursors for messaging
 - servers 21
 - supporting transformations 250
- debugging
 - handling common issues 358
 - integrations 357

- managing 351
- subscription PeopleCode 357
- transform programs 251
- transformations 492
- delegation, reusing connector code
 - via 549
- DER 391
- Destination Definition Wizard 371
- digital certificates
 - authenticating nodes 199
 - certificate authorities 389
 - See Also* CA
 - configuring integration gateways 35, 45
 - CSR 390
 - See Also* CSR
 - digital signatures 387
 - See Also* digital signatures
 - Distinguished Name (DN) 390
 - handling nonrepudiation messages 97
 - implementing nonrepudiation 150
 - installing for client authentication 412
 - installing for web server authentication
 - (BEA WebLogic) 401
 - (IBM WebSphere) 405
 - installing node-based 394
 - obtaining signed public encryption
 - keys 391
 - public/private encryption keys 390
 - setting up for server authentication on
 - application servers 409
 - understanding 387, 389
 - understanding for server
 - authentication 401
 - understanding installation 390
- Digital Certificates page 394
- digital signatures
 - digital certificates 387
 - See Also* digital certificates
 - generating 394
 - implementing authentication 400
 - understanding 387
- dispatchers
 - asynchronous publishing of
 - instances 295
 - asynchronous publishing of publication
 - contracts 296
 - asynchronous subscription
 - contracts 299

- asynchronous subscription of
 - instances 298
- changing dispatcher status for
 - processes 314
- clean-up mode 314
- dedicating 25
- messaging servers 22
- Publication Broker 294
 - See Also* broker dispatchers/handlers
- Publication Contractor 294
 - See Also* publication dispatchers
- setting parameters 28
- Subscription Contractor 294
 - See Also* subscription dispatchers
- throttling messages 417
- understanding 28, 294
- understanding clean-up mode 313
- viewing status 313
- Distinguished Encoding Rules (DER) 391
- Distinguished Name (DN) 390
- DN 390
- Document Object Model (DOM), *See* XML DOM
- documentation
 - printed xx
 - related xx
 - updates xx
- Domain Status page
 - activating/deactivating messaging server
 - domains 313
 - changing dispatcher status for
 - processes 314
 - setting domain grace periods 314
 - understanding 312
 - viewing dispatcher status 313
- domains
 - activating messaging server
 - domains 21, 313
 - deactivating messaging server
 - domains 314
 - Domain Status page 312
 - See Also* Domain Status page
 - modifying failover priorities 316
 - pausing 305
 - PeopleSoft Domain Administration
 - menu 22
 - See Also* PSADMIN
 - recovering from stalled channels 303
 - resetting 305
 - setting grace periods 314
 - setting up failover 305, 314
 - splitting channels 25, 203
 - taking offline 305
 - understanding dedicated messaging
 - servers 23
 - working with messaging server
 - domains 312
- E**
 - encoding strings 92
 - encryption
 - configuring integration gateways 35, 45
 - FTP target connector passwords 118
 - generating digital signatures 394
 - HTTP listening connectors 80
 - HTTP target connectors 83
 - integrationGateway.properties
 - passwords 44
 - JMS error queue passwords 107
 - JMS error topic passwords 107
 - JMS message header passwords 108
 - JMS queue listener passwords 105
 - JMS target connector passwords 113
 - JMS topic subscriber passwords 106
 - obtaining signed public encryption
 - keys 391
 - PeopleSoft 8.1 connectors 101
 - PeopleSoft 8.1 listening connectors 81
 - PeopleSoft 8.1 target connectors 83
 - PSCipher 41, 44
 - public/private keys 390
 - securing message parameters 95
 - selecting for public key certificates 395
 - simple file target connector
 - passwords 115
 - submitting SOAP messages 99
 - understanding 387
 - userGatewayProfile.xml password 41
 - endpoints
 - for published node transactions 384
 - understanding 362
 - enterprise resource planning (ERP)
 - connectors, *See* ERP connectors
 - Entrust Technologies 389
 - environment variables
 - %TransformData 251
 - ERP connectors
 - configuring 369
 - delivered with iWay SOAPswitch 363

- documentation 365
 - generating WSDL 368
 - installing 363
 - J2EE adapter 363
 - Oracle Applications Database (OAP) adapter 363
 - SAP R/3 adapter 364
 - Siebel adapter 364
 - XML adapter 364
 - ERP Connectors Admin page 367
 - errors
 - asynchronous publishing of publication contracts 296
 - asynchronous subscription contracts 299
 - building error handling into listening connectors 535
 - building error handling into target connectors 531
 - capturing JMS connector errors in topics 107
 - correcting data errors on pages 191
 - correcting for inbound messages 189
 - correcting message errors in PeopleCode 190
 - deleting channels during upgrade 143
 - deleting messages during upgrade 151
 - displaying message status 301
 - error handling service 8
 - integration gateways 351
 - JMS listening connectors 104
 - JMS target connectors 114
 - listening connectors 352
 - logging for integration gateways 48
 - logging for messages 50
 - logging methods and parameters 356
 - managing 351
 - managing gateway error logging 355
 - missing JMS message header properties 108
 - processing for inbound messages 188
 - public key certificates 397
 - queueing JMS listening connector errors 107
 - raising errors during PeopleCode steps 495
 - refreshing gateways 53
 - response message error codes 63, 64
 - running batch notifications 344
 - runtime transformations 53
 - setting allowable service failures for dispatchers 29
 - setting allowable service failures for handlers 31
 - setting message destinations in HTTP headers 96
 - setting up logging 354
 - stalled channels 303
 - submitting non-SOAP messages 99
 - submitting SOAP messages 99
 - target connectors 351
 - transformations 252
 - transmitting inbound messages 98
 - understanding logging 9, 353
 - validating inbound messages 187
 - viewing for asynchronous messages 341
 - viewing for synchronous messages 344
 - events
 - OnAckReceive 164
 - OnRequest 165, 184
 - OnRouteReceive 158, 171
 - OnRouteSend 163, 195, 208
 - OnSend 158, 169
 - PreBuild 190
 - routing PeopleCode 156
 - SavePostChange 190
 - Subscription 65, 159, 161
 - exceptions
 - integration gateways 352
 - Java 353
 - JMS target connectors 114
 - logging for integration gateways 48
 - standard 352
 - Extensible Markup Language (XML), *See* XML
 - external message IDs
 - HTTP listening connector 89
 - JMS listening connector 104
- ## F
- failover
 - modifying priorities 316
 - setting up for domains 305, 314
 - setting up for group domains 25
 - Failover Configuration page 314
 - fields
 - aliases 67
 - changes to field-level attributes 72
 - database record 141

- message header/XML 141
- File Transfer Protocol (FTP), *See* FTP
- filtering
 - messages 494
 - messaging information 305
 - PeopleCode filtering example 256
 - understanding 247, 255

- firewalls
 - communicating with third-party systems 437
 - integrating with Integration Broker systems 427
 - receiving third-party messages 442, 446
 - sending messages to third-party systems 440, 444

- FTP
 - FTPS communication 119
 - servers 116
 - target connectors 83
 - See Also* FTP target connectors
 - using ConnectorRequestURL 55

- FTP servers 116

- FTP target connectors
 - directory list support 119
 - required JAR files 117
 - setting compression properties 85
 - setting FTPS connector properties 119
 - setting properties 117
 - understanding 83, 116
 - working with non-XML files 76

- FTPS communication 119

G

- gateway manager
 - gateway services 8
 - integration gateway architecture 7
 - understanding 8

- gateway private keys
 - setting up (BEA WebLogic) 404
 - setting up (IBM WebSphere) 408

- Gateway Properties page 41

- gateway services 8

- Gateways component
 - accessing/editing
 - integrationGateway.properties 41
 - building introspection into target connectors 530
 - configuring security 35
 - defining gateways 422

- registering connectors 537
- setting file security 115
- setting target connector properties 84
- understanding 36

- Gateways page

- configuring load balancing 418
 - defining integration gateways 36
 - registering connectors 38

- generic routing

- configuring hubs 432
 - understanding 432

- GetMail target connectors 84, 123

- Getting Started With Integration Broker 1

- Global Unique Identifier (GUID) 343

- glossary 571

- GUID 343

H

- handlers

- asynchronous publishing of instances 295
 - asynchronous publishing of publication contracts 296
 - asynchronous subscription contracts 299
 - asynchronous subscription of instances 298
 - debugging 358
 - dedicating 25
 - messaging servers 22
 - Publication Broker 294
 - See Also* broker dispatchers/handlers
 - Publication Contractor 294
 - See Also* publication handlers
 - setting parameters 31
 - Subscription Contractor 294
 - See Also* subscription handlers
 - understanding 31, 205, 294

- headers

- developing JNI headers 546
 - header section for request messages 58
 - header section for response messages 62

- HTTP

- connectors 80
 - See Also* HTTP connectors

- headers 90

- See Also* HTTP headers

- HTTPS communication 80

- See Also* HTTPS communication

- status codes 99
 - See Also* HTTP status codes
 - using ConnectorRequestURL 55
 - HTTP connectors
 - adding nonrepudiation signatures 96
 - formatting messages 92
 - HTTP status codes 99
 - See Also* HTTP status codes
 - implementing SSL encryption 388
 - listening connectors 80
 - See Also* HTTP listening connectors
 - passing parameters to 94
 - responding to request messages 98
 - routing transactions through proxy servers 100
 - submitting messages inside XML wrappers 93
 - submitting non-XML messages 93
 - submitting SOAP messages 98
 - target connectors 83
 - See Also* HTTP target connectors
 - transmitting messages 92
 - understanding 87
 - HTTP headers
 - passing HTTP parameters 94
 - setting message destinations 96
 - submitting cookies 97
 - submitting SOAP messages 98
 - understanding IBInfo data 90
 - HTTP listening connectors
 - setting message parameters 87
 - understanding 80, 87
 - using external message IDs 89
 - HTTP status codes
 - submitting non-SOAP messages 99
 - submitting SOAP messages 99
 - HTTP target connectors
 - default remote gateway connector 202
 - encoding strings 92
 - HTTP headers 90
 - See Also* HTTP headers
 - setting compression properties 85
 - setting properties 85, 90
 - setting the content-type property 92
 - specifying nodes for services 376
 - understanding 83, 90
 - HTTPS communication
 - HTTP connectors 87
 - HTTP listening connectors 80
 - PeopleSoft 8.1 connectors 101
 - PeopleSoft 8.1 listening connectors 81
 - PeopleSoft 8.1 target connectors 83
 - hubs
 - configuring generic-routing 432
 - configuring sender-specific routing 434
 - integrating with Integration Broker systems 430
 - retaining messages at hub nodes 288
 - selecting nodes 200
 - understanding routing 275, 432
- I**
- IBInfo
 - accessing using PeopleCode 65
 - receiving messages via JMS listening connectors 104
 - receiving third-party messages 442, 446
 - retrieving data 176
 - understanding 169
 - understanding HTTP headers 90
 - understanding JMS headers 111
 - understanding request messages 57, 59
 - understanding response messages 63
 - IBM MQSeries, *See* MQSeries
 - IBM WebSphere, *See* WebSphere
 - IBRequest, *See* request messages
 - IBResponse, *See* response messages
 - Import WSDL page 373
 - Inactive Objects page 242
 - inbound
 - messaging 154
 - See Also* inbound messaging
 - transactions 154
 - See Also* inbound transactions
 - Inbound Asynchronous Post page 324
 - inbound messaging
 - correcting errors 189
 - handling asynchronous 171
 - handling cookies 168
 - handling synchronous 184
 - invoking error processing 188
 - iWay SOAPswitch operations 364
 - See Also* iWay SOAPswitch
 - monitoring synchronous messages 305
 - request message flow through the architecture 11
 - responding to request messages 98
 - routing events 156

- setting default Jolt connect string properties 47
 - transforming messages 276
 - understanding 154
 - validating data 187
 - viewing asynchronous post statistics 324
 - viewing performance statistics 318
 - viewing synchronous message statistics 329
- Inbound Synchronous page 329
- inbound transactions
 - adding systems 369
 - creating destinations/notifications 371
 - generating WSDL via ERP connectors 369
 - transaction modifiers 274
 - See Also* transaction modifiers
 - understanding 154
 - understanding asynchronous /synchronous 10
 - understanding routing 195
- inheritance, reusing connector code via 549
- Integration Broker
 - connector SDK 523
 - See Also* connector SDK
 - implementing 1
 - Monitor 291
 - See Also* Monitor
 - understanding xxv, 5
- Integration Broker Monitor, *See* Monitor
- integration engine
 - architecture 9
 - bypassing to send messages 54
 - understanding 6
- integration gateway
 - administering 36
 - applying transformations 50
 - See Also* transformations
 - architecture 6
 - bypassing the integration engine to send messages 54
 - configuring 36
 - configuring for load balancing 418
 - configuring security 35
 - connector management service 9
 - See Also* connectors
 - defining 36, 422
 - error handling 351
 - error handling service 8
 - gateway manager 8
 - See Also* gateway manager
 - gateway services 8
 - Gateways component 36
 - general configuration 36
 - handling common issues 358
 - integrationGateway.properties file 41
 - See Also*
 - integrationGateway.properties
 - local gateway compatibility 35
 - logging errors/messages 353
 - managing 35
 - managing error logging 355
 - managing message logging 354
 - message validation service 9
 - messaging objects service 8
 - refreshing properties 39
 - remote gateways 439
 - See Also* remote gateways
 - running HTTP transactions through proxy servers 100
 - setting security properties 45
 - setting up error/message logging 354
 - setting up logging 48
 - specifying for nodes 202
 - specifying gateway versions/class-locations 46
 - understanding 5
 - understanding exceptions 352
 - understanding gateway definitions 15
 - understanding SSL 388
 - viewing non-English characters in log files 354
 - XML parsing service 9
- Integration Point Viewer
 - accessing integrations 235
 - activating/viewing inactive objects 242
 - understanding 235
 - viewing integration information 236
- Integration Point Wizard
 - accessing relationships 232
 - creating integrations 216
 - overriding transaction routing type/connector properties 222
 - prerequisites 215
 - setting transaction logging /properties 224
 - specifying asynchronous reply message and transformation 226

- specifying external names for
 - messages 217
- specifying nodes 217, 218
- specifying receiver/sender message
 - information 219
- specifying transaction types 218
- specifying transformation
 - informations 220
- understanding 215, 216
- understanding transactions 217, 218
- understanding transformations 218
- viewing/saving integrations 227
- integrationGateway.properties
 - accessing 41
 - configuring security 45
 - connectors 46
 - See Also* connectors
 - controlling routing 534
 - enabling messaging system performance
 - statistics feature 322
 - encrypting passwords 44
 - refreshing properties 39
 - setting up logging 48, 354
 - specifying gateway versions/class-
 - locations 46
 - transforming messages at gateway 51
 - understanding 41, 422
- integrations
 - accessing 235
 - accessing relationships 232
 - applying transaction modifiers 15
 - asynchronous 21
 - See Also* asynchronous integrations
 - asynchronous-to-synchronous 217, 240
 - See Also* asynchronous-to-synchronous integrations
 - configuration scenarios 421
 - configuring nodes 196
 - debugging 357
 - integrating between nodes
 - (example) 499
 - integrating through a firewall 427
 - integrating via hubs 430
 - integrating with Integration Broker
 - systems 425
 - integrating with PeopleSoft 8.1x 448
 - integrating with third-party
 - systems 437, 439
 - See Also* remote gateways
 - Integration Point Viewer 235
 - See Also* Integration Point Viewer
 - integration points 215
 - See Also* Integration Point Wizard
 - relationships 245
 - See Also* relationships
 - setting up overview 16
 - synchronous 21
 - See Also* synchronous integrations
 - third-party 361
 - See Also* WSDL
 - transactions 204
 - See Also* transactions
 - understanding nodes 195
 - understanding setup 421
 - understanding transactions 195
- International Organization for
 - Standardization (ISO)
 - language codes 71
 - PeopleSoft timestamp formats 73
- introspection
 - building into target connectors 530
 - registering connectors via 38
 - scenario for target connector
 - development 528
- iPlanet
 - adding JAR files for JMS
 - connectors 103
 - setting JNDIFactory class names 103
- ISO
 - language codes 71
 - PeopleSoft timestamp formats 73
- iWay SOAPswitch
 - adding destinations 371
 - adding roles/systems/users 369
 - Administration Console 363
 - See Also* iWay SOAPswitch Administration Console
 - changing password/user ID 367
 - components 363
 - creating notifications 372
 - creating web services 370
 - default password/user ID 367
 - documentation 365
 - generating WSDL 368
 - importing WSDL 372
 - installing 363
 - logging in 367
 - MySQL database 364
 - notifications 364

- operation types 364
- SOAPswitch servers 363
 - See Also* iWay SOAPswitch server
- starting/stopping 366
- understanding 363
- understanding delivered adapters 363
 - See Also* ERP connectors
- web services 364
- Web Services Viewer 363
- iWay SOAPswitch Administration Console
 - accessing 368
 - understanding 363
- iWay SOAPswitch server
 - starting/stopping 366
 - understanding 363

J

- J2EE adapter 363
- JAR files
 - adding for JMS connectors 103
 - conflicting in WebSphere 103
 - required for FTP target connectors 117
- Java
 - exceptions 353
 - messaging service 110
 - See Also* JMS
 - native directory interface 103
 - See Also* JNDI
 - native interface 525
 - See Also* JNI
 - target connectors 545
 - See Also* Java target connectors
 - Xerces 545
 - XML DOM wrapper 542
 - 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++ 546
 - template for creating 546
 - understanding 545
- Java XML DOM wrapper
 - sample code 543
 - understanding 542
 - using wrapper classes 542
- JMS
 - listening connectors 81
 - See Also* JMS listening connectors

- providers 103
 - See Also* JMS providers
- queue listeners 104, 105
- setting header properties 108
 - See Also* JMS headers
- target connectors 83
 - See Also* JMS target connectors
- topic subscribers 104, 106
- understanding 110
- JMS connectors
 - JAR files 103
 - JMS provider support 103
 - listening connectors 81
 - See Also* JMS listening connectors
 - setting JNDIFactory class names 103
 - target connectors 83
 - See Also* JMS target connectors
 - understanding 102
- JMS headers
 - setting properties for listening connectors 108
 - understanding IBInfo data for target connectors 111
- JMS listening connectors
 - error handling 104
 - external message IDs 104
 - JMS headers 108
 - See Also* JMS headers
 - JMS queue listeners 104
 - JMS topic subscribers 104
 - receiving messages 104
 - setting error queue/topic properties 107
 - starting/stopping 109
 - understanding 81, 104
- JMS providers
 - setting for target connectors 113
 - setting JNDIFactory class names 103
 - supported 103
 - using JMS target connectors 110
 - See Also* JMS target connectors
- JMS queue listeners 104, 105
- JMS target connectors
 - errors and exceptions 114
 - setting compression properties 85
 - setting properties 112
 - synchronous/asynchronous communication 111
 - understanding 83, 110
 - understanding JMS headers 111
 - See Also* JMS headers

- verifying setup 114
- JMS topic subscribers 104, 106
- JNDI
 - JMS error queues 107
 - JMS error topics 108
 - JMS queue listeners 105
 - JMS topic subscribers 106
 - setting JNDIFactory class names 103
 - understanding JMS connectors 103
- JNI
 - creating headers 546
 - developing connectors 545
 - understanding 525
- Jolt
 - setting connect gateway properties 422
 - setting default connect string
 - properties 47
 - setting node-specific connect string
 - properties 48
 - understanding 83

K

- keystore
 - identifying the encryption key pair 45
 - implementing nonrepudiation 388
 - setting the filepath/filename
 - /password 45
 - understanding 387
 - understanding public/private encryption
 - keys 390
- keystores
 - importing signed private keys into
 - (BEAWebLogic) 404
 - importing signed private keys into (IBM
 - WebSphere) 407
- Keytool utility 413

L

- languages
 - selecting for transformations
 - /translations 250
 - understanding message language
 - codes 71
 - viewing non-English characters in log
 - files 354
- WSDL 361
 - See Also* WSDL
- XML 477
 - See Also* XML

- listening connectors
 - building error handling/logging
 - into 535
 - building servlet/non-servlet based 533
 - controlling message routing 533
 - developing 525
 - developing classes for 533
 - error handling 352
 - HTTP 80
 - See Also* HTTP listening connectors
 - installing classes 537
 - integrating with third-party
 - systems 437
 - invoking 533
 - JMS 81
 - See Also* JMS listening connectors
 - message logging 355
 - PeopleSoft 80, 86
 - PeopleSoft 8.1 81, 101
 - receiving requests 11, 14
 - receiving responses 14
 - replacing null characters 81
 - understanding 8, 79
 - understanding delivered 80
 - using templates 539
- load balancing 418
- local messaging 206
- Local Synchronous page 333
- logging
 - application servers 357
 - building into listening connectors 535
 - building into target connectors 531
 - enabling for transformations 492
 - error logging methods and
 - parameters 356
 - errors and messages 9
 - gateway refresh errors 53
 - integration gateway 48
 - iWay SOAPswitch 364
 - managing 351
 - managing for gateway messages 354
 - managing gateway error logging 355
 - message logging in connectors 354,
 - 355
 - message logging methods and
 - parameters 355
 - messaging errors 50
 - runtime transformation errors 53
 - setting for synchronous messages 211
 - setting up error/message logging 354

- synchronous message errors 344
- system messages 49
- transactions 224
- understanding error/message
 - logging 353
- viewing non-English characters 354

M

- MCGetMail target connectors 84

- menus

- PSADMIN 22

- See Also* PSADMIN

- Message Channel Properties dialog

- box 139

- Message class 159

- Message definition window 145

- Message Details component

- understanding/accessing 336

- viewing asynchronous message properties 337

- message format

- example of rowset-based message 73

- fieldtype sections 68

- MsgData sections 68

- PeopleSoft rowset-based message format 66

- rowset-based message template 67

- timestamp format 73

- Message Monitor component

- monitoring channel status 311

- pausing/testing/pinging nodes 310

- setting up domain failover 314

- viewing messaging performance statistics 316

- viewing undelivered node

- transactions 309

- Message Properties dialog box 148

- Message Properties page 337

- Message Subscription Properties dialog

- box 161

- Messages page 211

- messages, logging system 49

- messaging

- activating messages 149, 244, 278

- activating subscription

- PeopleCode 244

- archiving/retrieving message

- instances 336

- bypassing the integration engine 54

- channels 138

- See Also* channels

- compression 85

- See Also* compression

- configuring message properties 148

- configuring messaging servers for

- asynchronous messaging 21

- connectors 79

- See Also* connectors

- controlling message size 175

- controlling routing 533

- converting characters to uppercase 147

- creating channels for service messages

- (WSDL) 381

- creating messages for services

- (WSDL) 380

- creating subscriptions 161

- See Also* subscriptions

- creating/defining message versions 145

- deactivating messages 278

- defining channels 137

- defining messages 137, 143

- deleting messages during upgrade 151

- dispatchers 294

- See Also* dispatchers

- external message IDs 89

- extracting messages 477

- filtering 255

- See Also* filtering

- gateway manager 8

- See Also* gateway manager

- generating messages 162

- generating test messages 192

- generating WSDL from messages 384

- handlers 294

- See Also* handlers

- handling cookies 168

- handling non-XML data 252

- IBInfo data for HTTP headers 90

- See Also* IBInfo

- identifying field-level attribute

- changes 72

- implementing nonrepudiation 150, 400

- inbound 98

- See Also* inbound messaging

- making working storage data available

- globally 253

- managing logging 354

- messaging services 8

- modifying failover priorities 316

- monitoring information 291

- See Also* Monitor
- nodes 196
 - See Also* nodes
- non-rowset-based messages 137
- nonrowset-based messages 191
 - See Also* nonrowset-based messages
- PeopleSoft rowset-based message
 - format 66
 - See Also* message format
- prerequisites for message
 - delivery/reception 153
- processing messages 171
- processing messages in parallel 140
- purging messaging tables 348
- receiving messages 171
- renaming message versions 146
- replacing null characters 81
- request messages 57
 - See Also* request messages
- response messages 62
 - See Also* response messages
- resubmitting/cancelling 309
- retaining messages at hub nodes 288
- routing PeopleCode 156
- rowset-based messages 137, 191
 - See Also* rowset-based messages
- running batch archiving 347
- running message queries 335
- selecting channels for service messages (WSDL) 380
- selecting messages for services (WSDL) 378
- sending messages 162
- sending messages in order 140
- sending/receiving messages via PeopleCode 15
- servers 21
 - See Also* messaging servers
- setting a default message version 146, 150
- setting the Tuxedo queue size 32
- setting up domain failover 314
- setting up logging 354
- submitting SOAP messages 98
- throttling dispatched messages 417
- transaction definitions for local 206
- transaction types 10
- transformations 50
 - See Also* transformations
- translating 259

- See Also* translations
- understanding 137
- understanding logging 9, 353
- understanding message definitions 15
- understanding message
 - delivery/reception 153
- understanding process flows 154
- understanding security 387
 - See Also* security
- understanding SOAP compliance 75
- understanding supported message
 - structures 57
- using records in message
 - definitions 147
- validating messages 149
- viewing and correcting 150
- viewing performance statistics 316
- XML DOM compliance 75
 - See Also* XML DOM
- messaging PeopleCode, *See* PeopleCode
- messaging servers
 - activating domains 21, 313
 - administrating 21
 - assigning channels 23
 - configuring 24, 28
 - deactivating domains 313
 - deleting 28
 - dispatchers/handlers for 22
 - editing channel lists 27
 - managing via PSADMIN 22
 - See Also* PSADMIN
 - processes for asynchronous messages 294
 - Publication Broker 22
 - See Also* Publication Broker
 - Publication Contractor 22
 - See Also* Publication Contractor
 - setting domain grace periods 314
 - setting persistent cursors 21
 - Subscription Contractor 22
 - See Also* Subscription Contractor
 - understanding processes 22
 - using dedicated servers 23
 - working with dedicated servers 25, 26
 - working with domains 312
- metadata
 - creating transaction metadata for imported WSDL 375
 - generating/viewing service transaction metadata 381

methods

- addConnectorField 530
- Connect 533
- CopyRowsetDelta 71
- CopyRowsetDeltaOriginal 71
- CopyToRowset 254
- ExecuteEdits 187, 189
- extracting information from the
 - Monitor 349
- getDestinationSystem() 534
- GETDIRLIST 119
- InboundPublish 185
- logError 356, 530
- logMessage 355
- Ping 529
- Publish 163
- selecting a simple file connector method
 - for messages 116
- selecting an FTP method for
 - messages 117, 126
- selecting an HTTP method for
 - messages 91
- Send 529
- SyncRequest 130, 165, 184, 208
- Update 190
- ValidateSoapDoc 187

MIME

- request messages 57
- response messages 62
- understanding PeopleSoft listening
 - connectors 86

MMA Partners xx

Monitor

- components 292
- extracting information from 349
- filtering messaging information 305
- Monitor Message component 304
 - See Also* Monitor Message component
- monitoring asynchronous message
 - details 336
 - See Also* Message Details component
- monitoring synchronous message
 - details 341
 - See Also* Synchronous Details component
- permission lists 291
- purging messaging tables 348
- running batch archiving 347
- running batch error notifications 344
- security 291

- understanding 291, 292

Monitor Message component

- accessing 304
- filtering 305
- monitoring asynchronous message
 - instances 308
- monitoring publication contracts 308
- monitoring queue information 307
- monitoring subscription contracts 308
- monitoring synchronous message
 - instances 308
- pages within 305
- resubmitting/cancelling messages for
 - processing 309
- understanding 304
- viewing channel partitioning
 - information 311
- viewing output 307
- working with messaging server
 - domains 312

MQSeries

- adding JAR files for JMS
 - connectors 103
- setting JNDIFactory class names 103

MultiChannel Framework (MCF) GetMail

- target connectors 84

Multipurpose Internet Mail Extension

- standard (MIME), *See* MIME

MySQL database (for iWay

- SOAPswitch) 364

N

New Role Definition Wizard 370

Node Definitions - Transactions page 127

Node Definitions component

- Connectors page 202
- Contact/Notes page 201
- Node Definitions page 196
- Properties page 201
- Transactions page 206

Node Transactions page 207

nodes

- activating 198, 210, 244, 278
- activating nonrepudiation 199
- adding pause times 310
- applying transaction modifiers 15
- associating with codesets 490
- associating with transactions 207
- authenticating 199
- certificates, setting up 410

- configuring 196
- configuring local/remote 422
- configuring the POP3 default node 127
- converting request messages 275
- copying definitions 200
- creating definitions 196
- deactivating 210, 278
- debugging 358
- defining for services (WSDL) 377
- defining properties 201
- deleting definitions 203
- deleting pause times 310
- generating WSDL 384
- implementing nonrepudiation 388, 400
- installing node-based digital certificates 394
- installing remote certificates 398
- installing signed public key certificates 395
- integrating between nodes (example) 499
- Integration Point Wizard 215
 - See Also* Integration Point Wizard
- overriding connectors 210
- overriding initial/result nodes 266
- pausing/testing 310
- pinging 310, 311
- publishing system node name 171
- relationships 273
 - See Also* relationships
- renaming 201, 203
- representing with images 200
- resolving multiple synchronous targets 208
- retaining messages at hub nodes 288
- routing PeopleCode 156
- routing types 195, 199
- routing, overriding 209
- selecting a contact manager 201
- selecting codeset groups 200
- selecting for relationships 278
- selecting hub 200
- selecting types 199
- setting as local 198
- setting connector properties 203
- setting for transformations 52
- setting HTTP connector properties 91
- setting JMS target connector properties 112
- setting Jolt connect strings 47, 48, 422

- setting properties for JMS message headers 108
- setting target connector properties 84
- simulating receiving messages from external nodes 185
- specifying connectors/gateways 202
- specifying contact information 201
- specifying for integrations 218
- specifying for JMS queue listeners 105
- specifying for JMS topic subscribers 106
- specifying for services (WSDL) 376
- specifying receiving nodes 88
- testing local 311
- transaction modifiers 274
 - See Also* transaction modifiers
- transactions 204
 - See Also* transactions
- understanding 248
- understanding definitions 15
- understanding endpoints for published node transactions 384
- understanding external 199
- understanding ICType 199, 200
- understanding local and remote 196
- understanding PIA 199
- using the POP3 default node 128
- viewing asynchronous integration information 237
- viewing asynchronous-to-synchronous integration information 240
- viewing status 305
- viewing synchronous integration information 239
- viewing the master 200
- nonrepudiation
 - activating for nodes 199
 - compressing messages 85
 - configuring 400
 - digital certificates 387
 - See Also* digital certificates
 - generating digital signatures 394
 - implementing 150
 - processing messages 89
 - setting for messages 149
 - signatures 96
 - See Also* nonrepudiation signatures
 - understanding 387, 388
- nonrepudiation signatures
 - adding 96

- implementing 150
- viewing for asynchronous messages 341
- viewing for synchronous messages 342, 344
- nonrowset-based message format 75
 - generating WSDL 384
 - selecting messages for services (WSDL) 378
 - understanding 191
- nonrowset-based messages, *See* nonrowset-based message format
- notes xxii
- notifications
 - creating iWay SOAPswitch notifications 372
 - running batch error notification processes 344
 - understanding iWay SOAPswitch notifications 364

O

- OAP adapter 363
- objects
 - activating/viewing inactive 242
 - ConnectorInfo 169
 - IBInfo 169
 - See Also* IBInfo
 - Message 65, 191
 - messaging objects service 8
 - registering JMS-administered 112
 - Rowset 254
 - Simple Object Access Protocol (SOAP) 98
 - See Also* SOAP
 - TransformData 494
 - using ConnectorRequest function 54
 - XML parsing 9
 - XmlDoc 75, 254, 494
- OnAckReceive
 - accessing 159
- OnRequest
 - accessing 159
- OnRouteReceive
 - accessing 159
- OnRouteSend
 - accessing 159
- OnSend
 - accessing 159
- Oracle
 - monitoring subscription contracts 308
 - Oracle Applications Database (OAP) adapter 363
- Oracle Applications Database (OAP) adapter 363
- outbound
 - messaging 154
 - See Also* outbound messaging
 - transactions 154
 - See Also* outbound messaging
- outbound messaging
 - creating web services 370
 - handling asynchronous 163
 - handling cookies 168
 - handling synchronous 165
 - identifying SOAP faults 164
 - iWay SOAPswitch operations 364
 - See Also* iWay SOAPswitch
 - message class outbound
 - PeopleCode 163
 - overriding synchronous timeout interval 167
 - request message flow through the architecture 11
 - routing events 156
 - sending non-XML data 163
 - testing 162
 - transforming messages 250, 276
 - understanding 154, 162
 - understanding delivery order 162
 - viewing performance statistics 318
 - viewing synchronous message statistics 330
- Outbound Synchronous page 330
- outbound transactions
 - adding back-end systems 369
 - generating WSDL via ERP connectors 368
 - overriding connectors 203, 209, 210
 - overriding gateway selections 203
 - overriding the routing type 209
 - routing 199
 - transaction modifiers 274
 - See Also* transaction modifiers
 - understanding 154
 - understanding asynchronous /synchronous 10
 - understanding routing 195

P

- partitioning
 - enabling/disabling fields 140
 - selecting fields 141
 - setting partitioning subchannels 88, 108
 - understanding 140
 - understanding blocked channels 303
 - viewing information 311
- Password Encryption Utility 44
- passwords
 - accessing
 - integrationGateway.properties 41
 - authenticating messages 199
 - encrypting for target connectors 84
 - implementing authentication 399
 - iWay SOAPswitch 367
 - running HTTP transactions through proxy servers 100
 - selecting for transaction
 - connectors 212
 - setting for application server node 48
 - setting for FTP target connectors 118
 - setting for HTTP messages 88
 - setting for integration gateway 45
 - setting for iWay SOAPswitch 367
 - setting for JMS error queues 107
 - setting for JMS error topics 107
 - setting for JMS message headers 108
 - setting for JMS queue listeners 105
 - setting for JMS target connectors 113
 - setting for JMS topic subscribers 106
 - setting for keystores 45
 - setting for POP3 target connectors 125
 - setting for proxy authentication 91
 - setting for simple file target
 - connectors 115, 116
- pause times
 - adding to nodes 310
 - deleting from nodes 310
- PEM
 - certificate signing requests (CSR) 390
 - See Also* CSR
 - exporting/converting certificates 392
- PeopleBooks
 - ordering xx
- PeopleCode
 - accessing message data 251
 - built-in functions 54
 - See Also* built-in functions
 - classes 67
 - See Also* classes
 - correcting message errors 190
 - creating data for XSLT steps 495
 - creating message subscriptions 161
 - defining PSCAMA records 70
 - filtering 250
 - generating to terminate
 - transformations 272
 - identifying field-level attribute
 - changes 72
 - implementing codesets 488
 - making working storage data available globally 253
 - manipulating message format 93
 - Message class 159
 - methods 71
 - See Also* methods
 - objects 494
 - See Also* objects
 - raising errors 495
 - routing 156
 - sending/receiving 15
 - setting target connector properties 169
 - simulating receiving messages from
 - external nodes 185
 - SOAPDoc class 159
 - subscription 171
 - See Also* subscription PeopleCode
 - translation example 270
 - understanding message filtering 255
 - XMLDoc class 159
- PeopleCode built-in functions, *See* built-in functions
- PeopleCode, typographical
 - conventions xxi
- PeopleSoft 8.1
 - connectors 81
 - See Also* PeopleSoft 8.1 connectors
 - integrating 448
 - listening connectors 81, 101
 - target connectors 83
 - See Also* PeopleSoft 8.1 target connectors
- PeopleSoft 8.1 connectors
 - listening connectors 81, 101
 - target connectors 83
 - See Also* PeopleSoft 8.1 target connectors

- understanding 101
- PeopleSoft 8.1 listening connectors 81, 101
- PeopleSoft 8.1 target connectors
 - duplicate message exception 352
 - setting properties 102
 - understanding 83, 101
- PeopleSoft Application Designer 138
 - See Also* Application Designer
- PeopleSoft application fundamentals xix
- PeopleSoft common application message attributes (PSCAMA), *See* PSCAMA
- PeopleSoft connectors
 - listening connectors 80, 86
 - target connectors 83, 86
 - understanding 85
- PeopleSoft Integration Broker, *See* Integration Broker
- PeopleSoft listening connectors 80, 86
- PeopleSoft rowset-based message format 66
- PeopleSoft Server Administration utility (PSADMIN), *See* PSADMIN
- PeopleSoft target connectors 83, 86
- performance issues
 - applying transformations at integration gateway 50
 - archiving messages 347
 - editing messaging server channel lists 27
 - editing XSLT transformations 51
 - load balancing 418
 - setting the Tuxedo queue size 32
 - throttling dispatched messages 417
 - understanding dedicated messaging servers 23
 - viewing messaging statistics 305, 316
- permission lists, Integration Broker Monitor 291
- persistent cursors, setting 21
- pinging
 - dispatchers 30
 - external systems 527
 - nodes 310
 - Ping method 529
 - publication contracts 296
 - remote nodes 311
 - target nodes 87
- PKI 388
- planning
 - architecture 1
 - integrations 1
 - security 2
 - staff skills required 2
 - support 2
- POP3
 - default node 127
 - See Also* POP3 default node
 - servers 123
 - See Also* POP3 servers
 - target connectors 84
 - See Also* POP3 target connectors
- POP3 default node
 - configuring 127
 - using 128
- POP3 servers
 - retrieving specific messages 130
 - setting connector properties 125
 - understanding 123
 - using UIDL 123
- POP3 target connectors
 - configuring the default node 127
 - GetMail target connectors 123
 - overriding defined properties 135
 - passing properties to 130, 135
 - retrieving specific messages 130
 - services for 123, 126
 - setting properties 125
 - understanding 84, 123
 - using the default node 128
- Post Office Protocol (POP3), *See* POP3
- prerequisites xix
- printed documentation xx
- Privacy Enhanced Mail (PEM), *See* PEM
- private keys 402
 - See Also* gateway private keys
 - generating (BEA WebLogic) 402
 - generating (IBM WebSphere) 406
 - generating private and public key pairs for client authentication 413
 - implementing nonrepudiation 388
 - importing signed keys into keystores (BEA WebLogic) 404
 - importing signed keys into keystores (IBM WebSphere) 407
 - setting distinguished name values 390
 - understanding encryption keys 390
- proxy servers
 - routing HTTP transactions 100
 - routing through 90

- setting properties 100
- PSADMIN
 - configuring messaging servers 28
 - creating/assigning dedicated messaging servers 26
 - deleting messaging servers 28
 - editing messaging server channel lists 27
 - enabling messaging system performance statistics feature 322
 - setting compression 65
 - setting the Tuxedo queue size 32
 - throttling dispatched messages 417
 - understanding 22
- PSCAMA
 - defining records 70
 - understanding language codes 71
 - understanding message XML fields 142
 - understanding MsgData sections for messages 68
 - using audit action codes 71
- PSCipher 41, 44
- PT_EMAIL_POP3 127
- public key certificates
 - accessing properties 391
 - installing signed 395
 - obtaining signed 391
 - resolving root certificate mismatches 398
 - understanding errors 397
- public key infrastructure (PKI) 388
- public keys
 - CSR 390
 - See Also* CSR
 - digital certificates 387
 - See Also* digital certificates
 - generating (BEA WebLogic) 402
 - generating (IBM WebSphere) 406
 - generating private and public key pairs for client authentication 413
 - importing (BEA WebLogic) 402
 - importing (IBM WebSphere) 406
 - infrastructure (PKI) 388
 - nonrepudiation 388
 - See Also* nonrepudiation
 - obtaining signed certificates 391
 - See Also* public key certificates
 - remote certificates 398
 - See Also* remote certificates
 - root certificates 391
 - See Also* root certificates
 - understanding encryption keys 390
- publication
 - asynchronous message publication 295
 - dispatchers 294
 - See Also* publication dispatchers
 - handlers 294
 - See Also* publication handlers
 - Publication Broker 293
 - See Also* Publication Broker
 - Publication Contractor 293
 - See Also* Publication Contractor
 - publication contracts 293
 - See Also* publication contracts
 - synchronous message publication 300
 - understanding blocked channels 303
 - understanding PubIDs 171
- Publication Broker
 - dispatchers/handlers 22
 - server processes 294
 - understanding 293, 295
- Publication Contract Handler page 326
- Publication Contractor
 - asynchronous message publication 295
 - dispatchers/handlers 22
 - server processes 294
 - understanding 293
- publication contracts
 - asynchronous publishing 296
 - monitoring 308
 - Pub Contracts page 305
 - setting properties 339
 - understanding 293
 - viewing information 339
 - viewing statistics 326
- publication dispatchers
 - asynchronous publishing of instances 295
 - asynchronous publishing of publication contracts 296
 - server process 294
- publication dispatchers/handlers
 - understanding 318
- publication handlers
 - asynchronous publishing of publication contracts 296
 - server process 294
 - viewing statistics 326
- Published EIPs page

- accessing/understanding 383
- generating from messages 384
- generating WSDL from component interfaces 384

Q

queries

- generating message detail reports 305, 335
- introspection 530
 - See Also* introspection
- running message queries 335
- transmitting URL query strings 96

Queries page 335

queues

- JMS queue listeners 104
- monitoring messaging information 304, 307
- refreshing messaging queues 30
- setting dispatcher properties 29
- setting error queue properties 107
- setting JMS target connectors timeouts 114
- Tuxedo 29
 - See Also* Tuxedo
- understanding blocked channels 303
- understanding messaging queues 293

R

records

- changing underlying definitions 147
- creating aliases 148
- defining PSCAMA records 70
- inserting into message definitions 147
- preserving aliases 254
- repositioning in message versions 148
- understanding aliases 67
- understanding message record structure 147

recycle count

- setting for dispatchers 29
- setting for handlers 31

refreshing

- gateway properties 39
- gateway refresh errors 53
- messaging queues 30

registration

- DLLs for native library 546
- JMS-administered objects 112

- target connectors 38, 537

related documentation xx

Relationship component 232

Relationship Contact page 279

Relationship Properties page 279

Relationship Transactions page 281

relationships

- activating 245, 278
- associating transformations with transactions 479
- changing status 278
- configuring 277
- deactivating 278, 280
- defining 277
- deleting 280
- determining parameters 275
- overriding node properties/contact information 279
- selecting nodes 278
- selecting status 278
- selecting transaction type combinations 275
- transaction modifiers 274
 - See Also* transaction modifiers
- understanding 273
- when to define 273
- where to define 276

Relationships — Nodes page 277

Relationships — Trans Modifiers page 281

remote certificates

- importing/exporting 395
- installing 398
- setting up 411

remote gateways

- changing the default connector setting 422
- configuring nodes 422
- integrating with Integration Broker systems 427
- integrating with third-party systems 439
- receiving third-party messages 441, 445
- sending messages to third-party systems 440, 444
- specifying for nodes 202

remote procedure call (RPC) 363

request messages

- content section 58, 62

- converting asynchronous to synchronous 275
- creating for services (WSDL) 380
- exceptions 353
- header section 57, 58
- IBInfo section 57, 59
- incoming flow through the architecture 11
- Integration Point Wizard 217
 - See Also* Integration Point Wizard
- internal format 57
- outgoing flow through the architecture 11
- selecting for services (WSDL) 378
- selecting for transactions 208
- specifying receiver message information 222
- specifying versions/external names 219
- transaction modifiers 274
 - See Also* transaction modifiers
- transforming 220, 250
- Request New Certificate page 396
- response messages
 - catalog entries 64
 - content section 64
 - creating for services (WSDL) 380
 - error codes 64
 - exceptions 353
 - header section 62
 - IBInfo section 63
 - Integration Point Wizard 217
 - See Also* Integration Point Wizard
 - internal format 62
 - selecting for services (WSDL) 378
 - selecting for transactions 212
 - specifying versions/external names 219
 - transforming 220
- ReturnToServer 157
- root certificates
 - accessing certificate properties 391
 - adding 410
 - exporting/converting 392
 - importing signed certificates 414
 - installing 395
 - installing node-based digital certificates 394
 - obtaining signed certificates 414
 - resolving mismatches 398
 - understanding 391
- routing

- applying basic 288
- controlling 533
- default connector 422
- explicit 195
- generic 432
 - See Also* generic routing
- hub 275
- implicit 195
- overriding 222, 223, 224
- PeopleCode 156
- Publication Broker 293
- selecting types 195, 199, 209
- sender-specified 432
 - See Also* sender-specified routing
- transaction modifiers 274
 - See Also* transaction modifiers
- routing events
 - accessing 159
 - OnAckReceive 159
 - OnRequest 159
- rowset-based message format 66, 137
 - selecting messages for services (WSDL) 378
 - understanding 191
- RPC 363
- Run Archive page 347

S

- SAP R/3 adapter 364
- Secure Sockets Layer (SSL), *See* SSL
- security
 - adding roles 370
 - authentication 387
 - See Also* authentication
 - certificate authorities (CA) 389
 - See Also* CA
 - configuring integration gateways 45
 - digital certificates 387
 - See Also* digital certificates
 - digital signatures 400
 - See Also* digital signatures
 - Distinguished Name (DN) 390
 - encryption 387
 - See Also* encryption
 - FTP target connectors 83
 - HTTP listening connectors 80
 - HTTP target connectors 83
 - Integration Broker Monitor 291
 - message channels 142
 - messaging security types 387

- nonrepudiation 96
 - See Also* nonrepudiation
- PeopleSoft 8.1 connectors 101
- planning for 2
- running batch error notifications 344
- sending message parameters 95
- setting properties 45
- setting properties for FTPS
 - communication 119
- simple files 115
- submitting SOAP messages 99
- transmitting URL query strings 96
- understanding 387
- sender-specified routing
 - configuring hubs 434
 - understanding 432
- server authentication
 - understanding 401
- servers
 - FTP 116
 - iWay SOAPswitch 363
 - See Also* iWay SOAPswitch server
 - messaging 21
 - See Also* messaging servers
 - POP3 123
 - See Also* POP3 servers
 - proxy 100
 - See Also* proxy servers
- services, gateway, *See* gateway services
- setting up integrations overview 16
- Siebel adapter 364
- signatures
 - digital 387
 - See Also* digital signatures
 - nonrepudiation 96
 - See Also* nonrepudiation signatures
 - using line feeds 97
- simple file target connectors
 - setting compression properties 85
 - setting file-security/properties 115
 - understanding 84, 115
- Simple Mail Transfer Protocol (SMTP)
 - target connectors, *See* SMTP target connectors
- single signon 199
- SMTP target connectors
 - setting compression properties 85
 - setting properties 122
 - understanding 83, 122
- SOAP
 - defining PSCAMA records 70
 - enabling access for third-party systems 92
 - identifying faults 164
 - iWay SOAPswitch 363
 - See Also* iWay SOAPswitch
 - setting message destinations in HTTP headers 96
 - status codes for SOAP messages 99
 - submitting SOAP messages 98
 - understanding error codes 64
 - understanding SOAP-compliant messages 75
 - using HTTP target connectors 90
- SOAPDoc class 159
- Solaris 308
- Specify Async Reply Message and Transformation page 226
- Specify Request Connector Properties page 224
- SSL
 - configuring integration gateways 35
 - encrypting integration gateways 45
 - encrypting message parameters 95
 - generating digital signatures 394
 - HTTP listening connectors 80
 - HTTP target connectors 83
 - PeopleSoft 8.1 connectors 101
 - PeopleSoft 8.1 listening connectors 81
 - PeopleSoft 8.1 target connectors 83
 - submitting SOAP messages 99
 - understanding 388
- SSL encryption
 - setting up 401
 - setting up for BEA WebLogic 405
 - setting up for IBM WebSphere 408
- Statistics page
 - Selecting statistics data to view 323
 - using the interface 321
 - viewing messaging system performance statistics 317
- status
 - changing dispatcher status for processes 314
 - changing for relationships 278
 - codes for response messages 63, 64
 - selecting channel status 139
 - selecting for relationships 278
 - selecting for transaction modifiers 285
 - selecting for transactions 206, 209

- setting for transformations 252
- understanding message status 301
- viewing dispatcher status 313
- viewing for channels 305, 311
- viewing for domains/nodes 305
- Sub Channel Message Queue page 311
- subchannels
 - applying partitioning 88, 108, 140
 - setting for JMS queue listeners 105
 - setting subchannels for JMS topic subscribers 106
 - understanding blocked channels 303
 - viewing partitioning information 311
- subscription
 - asynchronous subscription of
 - instances 298
 - contracts 293
 - See Also* subscription contracts
 - creating message subscriptions 161
 - dispatchers 294
 - See Also* subscription dispatchers
 - handlers 294
 - See Also* subscription handlers
 - PeopleCode 171
 - See Also* subscription PeopleCode
 - Subscription Contractor 293
 - See Also* Subscription Contractor
 - synchronous messaging 300
 - understanding 293
- Subscription Contract Handler page 325
- Subscription Contractor
 - dispatchers/handlers 22
 - server processes 294
 - understanding 293
- subscription contracts
 - asynchronous subscription of
 - messages 298
 - monitoring subscription contracts 308
 - setting properties 340
 - understanding 293
 - understanding asynchronous 299
 - viewing information 340
 - viewing work orders 305
- subscription dispatchers
 - asynchronous subscription
 - contracts 299
 - asynchronous subscription of
 - instances 298
 - server process 294
- subscription dispatchers/handlers
 - understanding 318
- Subscription event
 - accessing 159
- subscription handlers
 - asynchronous subscription
 - contracts 299
 - server process 294
 - viewing statistics 325
- subscription PeopleCode
 - activating 244
 - asynchronous subscription
 - contract 299
 - debugging 357, 358
 - understanding 171
- suggestions, submitting xxiii
- Sun iPlanet, *See* iPlanet
- support, planning for 2
- Sync Message Detail page 342
- synchronous
 - integrations 21
 - See Also* synchronous integrations
 - messaging/transmission 21
 - See Also* synchronous messaging
 - transactions 10
 - See Also* synchronous transactions
- Synchronous Details component
 - accessing/understanding 341
 - viewing message details 342
 - viewing message XML content 343
- synchronous integrations
 - configuring messaging servers 21
 - defining messages 137
 - understanding 217
 - viewing integration information 239
- synchronous messaging
 - applying transformations at gateway 51
 - converting asynchronous messages to
 - synchronous 275
 - debugging transformations 358
 - displaying status 301
 - handling cookies 168
 - hub routing 275
 - monitoring inbound messages 305
 - monitoring instances 308
 - monitoring message details 341, 342
 - See Also* Synchronous Details component
 - publishing 300
 - responding to request messages 98
 - routing events 156

- submitting non-SOAP messages 99
- transforming messages 250, 276
- understanding compression 65
- understanding message definitions 15
- understanding subscription 300
- using JMS target connectors 111
- using record/field aliases 67
- viewing attributes in XML 342
- viewing errors 344
- viewing inbound message statistics 329
- viewing local statistics 333
- viewing message content in XML 343
- viewing outbound message statistics 330
- viewing performance statistics 305, 316, 318
- viewing signature information 344
- synchronous transactions
 - accounting for multiple targets 208
 - asynchronous-to-synchronous transactions 217
 - See Also* asynchronous-to-synchronous transactions
 - handling inbound 184
 - handling outbound 165
- Integration Point Wizard 219
 - See Also* Integration Point Wizard
- iWay SOAPswitch operations 364
 - See Also* iWay SOAPswitch
- selecting channels for service messages (WSDL) 380
- selecting messages for services (WSDL) 378
- setting logging options 211
- setting properties 224
- SOAP compliance 75
- specifying receiver message information 222
- specifying sender message information 219
- transaction modifiers 274
 - See Also* transaction modifiers
- transforming messages 220
- understanding inbound 10
- understanding integrations 217
- understanding outbound 10
- viewing integration points 229
- System Definition Wizard, adding systems 369
- system messages, logging 49

T

- tables, purging messaging 348
- Target Connector Interface (TCI) 351, 529
- target connectors
 - building error handling/logging into 531
 - building introspection into 530
 - compression 65
 - creating in C/C++ 546
 - developing 525
 - developing classes for 529
 - development infrastructure 525
 - editing properties 39
 - encrypting passwords 44, 84
 - error handling 351
 - exceptions 352
 - FTP 83
 - See Also* FTP target connectors
 - GetMail 84
 - handling introspection 528
 - HTTP 83
 - See Also* HTTP target connectors
 - installing classes 537
 - Java 545
 - See Also* Java target connectors
 - JMS 83
 - See Also* JMS target connectors
 - message logging 354
 - overriding for transactions 210
 - overriding properties at runtime 169
 - overriding properties via OnSend 169
 - PeopleSoft 8.1 83
 - See Also* PeopleSoft 8.1 target connectors
 - PeopleSoft target connectors 83, 86
 - pinging external systems 527
 - POP3 84
 - See Also* POP3 target connectors
 - processing requests 11
 - receiving requests 14
 - receiving responses 13, 14
 - receiving third-party messages 441, 445
 - registering 38, 537
 - removing properties 203
 - selecting 422
 - sending messages to third-party systems 440, 444

- setting compression properties 85
- setting connector properties for nodes 203
- setting default Jolt connect string properties 47
- setting node-specific Jolt connect string properties 48
- setting properties 84, 85
- setting properties at runtime 169
- setting properties via OnSend 169
- simple file 84
 - See Also* simple file target connectors
- SMTP 83
 - See Also* SMTP target connectors
- specifying for nodes 202
- understanding 8, 81
- understanding delivered 82
- using templates 538
- using the target connector interface 351, 529
- TCI 351, 529
- templates
 - Java target connector 546
 - MsgData 69
 - rowset-based message format, PeopleSoft 67
 - sample XSLT template 508
 - using connector templates 538
- terms 571
- testing
 - Create Test Message feature 252
 - generating test messages 192
 - local nodes 311
 - nodes 310
 - outbound messages 162
- Thawte 389
- third-party
 - applications 80, 86
 - integrations 361
 - See Also* WSDL
 - load balancing software 418
 - message formatting/transmission 92
 - products 5
 - systems 92
 - See Also* third-party systems
 - XSLT tools 492
- third-party messaging
 - including FieldTypes sections in messages 68
 - integrating with third-party systems 437
 - receiving messages 441, 445
 - sending messages to third-party systems 440, 444
 - transforming messages 248
- third-party systems
 - configuring for integration with PeopleSoft 443, 447
 - integrating via remote gateways 439
 - integrating with 437
 - receiving SOAP transactions 92
- timeout
 - setting for handlers 31
- timeout interval, overriding for synchronous transactions 167
- timeouts
 - adding pause times to nodes 310
 - setting for FTP target connectors 118
 - setting for HTTP connectors 92
 - setting for JMS target connectors 114
 - setting for PeopleSoft 8.1 connectors 102
- tracing
 - application servers 357
 - managing 351
 - sample .AET trace file 493
 - transform programs 251
 - transformations 492
- transaction connectors 212
- Transaction Detail page 209
- Transaction Modifier Detail page 284
- transaction modifiers
 - activating 286
 - applying asynchronous to synchronous 287
 - applying at nodes 287
 - applying multiple 277
 - deactivating 286
 - defining 281
 - deleting 289
 - deleting/renaming transform programs 251
 - editing details 284
 - hub routing 275
 - managing 281
 - retaining messages at hub nodes 288
 - selecting initial transactions 283
 - selecting transaction/transmission types 275

- transforming messages 276
- understanding 274
- understanding connector overrides 283
- viewing 281
- transactions
 - activating 206, 209, 210, 244
 - associating with transformations 479, 513
 - asynchronous 10
 - See Also* asynchronous transactions
 - asynchronous-to-synchronous 217
 - See Also* asynchronous-to-synchronous transactions
 - configuring 204
 - configuring ERP connectors 369
 - copying transactions between nodes 207
 - creating transaction metadata for imported WSDL 375
 - cross-node messaging 205
 - deactivating 206, 209, 210
 - defined 15
 - defining 207
 - deleting 207
 - editing connector properties 212
 - editing details 209
 - editing message details 211
 - generating/viewing service transaction metadata 381
 - inbound 10
 - See Also* inbound transactions
 - Integration Point Wizard 217
 - See Also* Integration Point Wizard
 - local messaging 206
 - modifiers 274
 - See Also* transaction modifiers
 - outbound 10
 - See Also* outbound transactions
 - overriding connector properties 222, 223, 224
 - overriding routing types 222, 223, 224
 - relationships 273
 - See Also* relationships
 - selecting connectors 210
 - selecting request messages 208
 - selecting response messages 212
 - selecting types 208, 275
 - setting external message names 212
 - setting logging options 224
 - setting properties 224
 - specifying sender message information 219
 - specifying transaction types for integrations 218
 - specifying transformation information 220
 - synchronous 10
 - See Also* synchronous transactions
 - types 10
 - understanding 205, 218
 - understanding endpoints for published node transactions 384
 - viewing 206
 - viewing undelivered node transactions 309
- transform programs, *See* transformations
- transformation engine
 - associating nodes with codesets 490
 - filtering messages 494
 - psft_function tag 486
 - sample XSLT template 508
 - supporting callouts 495
 - understanding 478
 - using non-XML data 493
 - using the value tag 487
- transformations
 - accessing message data 251
 - accessing relationships 232
 - applying at integration gateway 50
 - applying to request/response messages 286
 - associating with transactions 479, 513
 - bypassing Integration Broker 498
 - combining with translations 250
 - See Also* translations
 - creating data for XSLT steps 495
 - creating transform programs 478
 - debugging 358, 492
 - defined 16
 - defining transform programs 249
 - deleting transform programs 251
 - developing 51, 477
 - enabling logging 492
 - handling non-XML data 252
 - including filtering 255
 - integrating between nodes (example) 499
 - integrating Integration Broker systems via hubs 430
 - Integration Point Wizard 218

- See Also* Integration Point Wizard
- invoking 250
- making working storage data available
 - globally 253
- output after first XSLT pass
 - (example) 514
- output after second codeset pass
 - (example) 518
- preserving record and field aliases 254
- raising errors during PeopleCode steps 495
- renaming transform programs 251
- setting properties for gateway-based 52
- setting up (example) 504
- specifying for asynchronous transactions 226
- specifying receiver message information 222
- subscription contracts 340
- terminating 272, 528
- tracing transform programs 251
- transaction modifiers 274
 - See Also* transaction modifiers
- transformation engine 478
 - See Also* transformation engine
- transforming messages 276
- transforming non-XML data 493
- transforming publication contracts 339
- transforming request/response messages 220
- transforming third-party messages 248
- understanding 247, 250, 257
- understanding runtime errors 53
- understanding XML 490
 - See Also* XML
- understanding XSLT 492
 - See Also* XSLT
- using PeopleCode 250, 488
- using TransformData class 252
- using TransformData object 494
- using XSLT 51, 250, 258, 485
- translations
 - codeset repository 260
 - See Also* codeset repositories
 - codesets 259
 - See Also* codesets
 - combining with transformations 250
 - developing 260
 - PeopleCode example 270
 - searching for data to translate 200

- understanding 247, 259, 479
- using the parm tag 266
- using the psft_function tag 265
- using the value tag 266
- using XSLT 250, 265
 - XSLT example 267
- transmission
 - asynchronous 111
 - See Also* asynchronous messaging
 - synchronous 111
 - See Also* synchronous messaging
- transmission, message
 - changing via transaction modifiers 276
 - complying with requirements 92
 - converting asynchronous to synchronous 275
 - mixing via transaction modifiers 205
 - responses to successful/failed 98
 - selecting types 275
- Tuxedo
 - configuring queue size 25, 32
 - setting the queue check frequency for dispatchers 29
 - throttling dispatched messages 417
- typographical conventions xxi

U

- UDDI interactions 75
- UIDL
 - passing connector properties as defined 130
 - retrieving specific messages from POP3 server 130
 - using POP3 services 123
 - using the POP3 default node 129
- Unique ID Listing (UIDL), *See* UIDL
- Universal Description, Discovery, and Integration (UDDI) interactions 75
- upgrade issues
 - assigning target connectors to nodes 423, 450
 - deleting message channels 143
 - deleting messages 151
 - local gateway compatibility 35
 - pause time considerations 310
 - renaming node definitions 203
- URL query strings 96
- URLs
 - debugging 358

- receiving third-party messages 442, 446
- setting for HTTP connectors 85
- URL query strings 96
- User Profile Definition page
 - defining messages 143
 - field list view 145
 - message structure view 144

V

- validation
 - authenticating nodes 199
 - channel names 25
 - inbound messages 187
 - setting for messages 149
 - understanding message validation 9
- Verisign 389
- visual cues xxii

W

- W3C
 - adding nonrepudiation signatures 96
 - documentation for HTTP headers 91
 - understanding WSDL terminology 362
 - XSLT 250
 - See Also* XSLT
- warnings xxii
- warnings, logging 48, 354
- web servers
 - WebLogic 103
 - See Also* WebLogic
 - WebSphere 103
 - See Also* WebSphere
- web services
 - creating 370
 - iWay SOAPswitch 363, 364
 - See Also* iWay SOAPswitch
 - understanding 362
 - Web Services Viewer 363
- Web Services Definition Wizard 370
- Web Services Description Language (WSDL), *See* WSDL
- WebLogic
 - adding JAR files for JMS connectors 103
 - generating CSRs 402
 - generating private keys 402
 - generating public keys 402
 - implementing SSL encryption 388

- importing public keys 402
- importing signed private keys into keystores 404
- installing digital certificates (server authentication) 401
- setting JNDIFactory class names 103
- setting up gateway private keys 404
- setting up SSL 405
- submitting CSRs to CAs for signing 404
- WebSphere
 - conflicting JAR files 103
 - generating CSRs 406
 - generating private keys 406
 - generating public keys 406
 - implementing SSL encryption 388
 - importing public keys 406
 - importing signed private keys into keystores 407
 - installing digital certificates (server authentication) 405
 - JMS provider support 103
 - setting up gateway private keys 408
 - setting up SSL 408
 - submitting CSRs to CAs for signing 407
- World Wide Web Consortium (W3C), *See* W3C
- WSDL
 - completing integrations 383
 - creating transaction metadata 375
 - See Also* WSDL Operation Wizard
 - creating WSDL documents 370
 - generating via ERP connectors 368
 - generating via messages/component interfaces 383
 - See Also* Published EIPs page
 - importing 372, 373
 - terminology 362
 - understanding bindings 363
 - understanding endpoints for published node transactions 384
 - understanding third-party integrations 361
 - using the repository 374
- WSDL Information page 372
- WSDL Operation Wizard
 - accessing 376
 - creating channels for service messages 381

- creating request/response messages for services 380
- defining nodes for services 377
- generating/viewing transaction metadata 381
- interface/understanding 375
- selecting channels for service messages 380
- selecting request/response messages for services 378
- specifying nodes for services 376
- WSDL Repository page 374

X

- Xalan 250, 545

- Xerces 545

XML

- adapter (iWay SOAPswitch) 364
- adding nonrepudiation signatures 96
- attributes/documents/namespaces /nodes 490
- DOM-compliant messages 75
 - See Also* XML DOM
- downloading/uploading XML for messages 339
- editing asynchronous message XML 341
- editing for publication contracts 339
- editing for subscription contracts 340
- extracting messages 477
- Java XML DOM wrapper 542
 - See Also* Java XML DOM wrapper
- message XML fields 142
- nonrowset-based messages 191
 - See Also* nonrowset-based messages
- parsers 9, 492
- PeopleSoft XML message wrapper 93
 - See Also* CDATA
- printing asynchronous message XML 341
- retrieving XML representations of messages 478
- rowset-based messages 191
 - See Also* rowset-based messages
- saving messages in XML 84
- sending nonrowset-based XML messages 15
- SOAP-specific errors 99
- understanding 490
- using built-in functions 191

- using record/field aliases 67
- using Xalan and Xerces 545
- viewing asynchronous messages 341
- viewing for publication contracts 339
- viewing for subscription contracts 340
- viewing sample message XML 252
- viewing synchronous message attributes 342
- viewing synchronous message content 343
- web services 362
 - See Also* web services

- WSDL documents 361

- See Also* WSDL

- XSLT 492

- See Also* XSLT

XML DOM

- handling non-XML data 252
- Java wrapper 542
 - See Also* Java XML DOM wrapper
- SOAP-compliant messages 75
- understanding 75
- validating messages 187

- XML Message Viewer page 341

- XMLDoc class 159

- XPath 487, 492

XSLT

- accessing message data 251
- creating data for XSLT steps 495
- implementing codesets 485
- manipulating message content 93
- sample template 508
- tags 265
 - See Also* XSLT tags
- transformations 51
 - See Also* transformations
- translations 250
 - See Also* translations
- understanding 492

XSLT tags

- name 487
- parm 266
- psft_function 265, 486
- value 266, 487

