



PeopleTools 8.12 PeopleSoft Application Messaging PeopleBook

PeopleTools 8.12 PeopleSoft Application Messaging PeopleBook

SKU MTAMr8SP1B 1200

PeopleBooks Contributors: Teams from PeopleSoft Product Documentation and Development.

Copyright © 2001 by PeopleSoft, Inc. All rights reserved.

Printed in the United States of America.

All material contained in this documentation is proprietary and confidential to PeopleSoft, Inc. and is protected by copyright laws. 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, Inc.

This documentation is subject to change without notice, and PeopleSoft, Inc. 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, Inc. in writing.

The copyrighted software that accompanies this documentation 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 documentation, including the disclosure thereof.

PeopleSoft, the PeopleSoft logo, PeopleTools, PS/nVision, PeopleCode, PeopleBooks, Vantive, and Vantive Enterprise are registered trademarks, and *PeopleTalk* and "People power the internet." are trademarks of PeopleSoft, Inc. All other company and product names may be trademarks of their respective owners.

Contents

About This PeopleBook

Before You Begin	xi
Related Documentation	xii
Documentation on the Internet	xii
Documentation on CD-ROM	xii
Hardcopy Documentation	xii
Typographical Conventions and Visual Cues	xiii
Comments and Suggestions	xiv

Chapter 1

Introduction to Application Messaging

Overview	1-1
Message Designers and Objects	1-1

Chapter 2

Designing Application Messages

Defining Message Nodes	2-1
Defining Message Channels	2-3
How Channels Are Used at Run Time	2-7
Setting Message Channel Properties	2-7
Adding Messages to Message Channel	2-9
Channel Partitioning	2-9
Defining Application Messages	2-11
Message Designer Views	2-11
Message Structure View	2-11
List View	2-11
Creating a New Message Version	2-12
Default Version Names	2-12
Default Message Structure Versions	2-12
Inserting a Record into a Message Version	2-13
Inserting a Child Record	2-13
Inserting a Message Subscription	2-14
View Subscription PeopleCode	2-15

Example of Single Node Publish and Subscribe	2-16
Viewing and Changing Message Properties.....	2-17
Creating a Test Message	2-18
Cross-node Messaging.....	2-19
Setup for Cross-node Messaging.....	2-20

Chapter 3

Publishing Messages

Design the Message Application	3-1
Define the Message Nodes	3-2
Define the Message Channel	3-2
Set the Channel Routings	3-3
Low-level Channel Routing	3-3
Define the Messages	3-3
PSCAMA Table	3-3
PSCAMA Record.....	3-4
Checking the Audit Action.....	3-6
Add Publish PeopleCode	3-8
Where to Put Your PeopleCode	3-9
PeopleCode Statements to Publish.....	3-9
Sample Publish PeopleCode	3-9
Tips on Publish PeopleCode	3-10
Test	3-10

Chapter 4

Subscribing to Messages

Design the Message Application	4-1
Define the Message Nodes	4-1
Define the Message Channel	4-2
Define the Messages	4-2
Add Subscribe PeopleCode	4-2
Sample Subscription PeopleCode	4-2
Sample 1.....	4-2
Sample 2.....	4-3
Checking Maximum Message Size	4-5
Test	4-7

Chapter 5

Processing Messaging Errors

Validating Data.....	5-1
----------------------	-----

Exit Function	5-2
Exit ()	5-2
Sample Exit () PeopleCode	5-3
Exit (1)	5-3
Sample Exit (1) PeopleCode	5-3
Correcting Errors.....	5-3
Correcting Message Errors in PeopleCode	5-4
Correcting Message Errors in the Message Monitor.....	5-6
Correcting Data Errors in a Page.....	5-6

Chapter 6

Third Party Integration

Application Messaging Gateway	6-1
Application Messaging SDK	6-1
SDK Resources	6-2
Publishing to PeopleSoft.....	6-2
Steps to Publish to PeopleSoft	6-3
Creating Subscription Message Components.....	6-3
Modifying the Handler.....	6-3
Creating Your Message.....	6-3
PeopleSoft XML Poster	6-4
PSXmlPost.exe.....	6-4
PSXmlPost.jar	6-5
PSXmlPost Parameters.....	6-7
Before You Publish.....	6-8
Using Publication IDs	6-9
Converting the XML File to UTF-8	6-9
Sample UTF-8 XML	6-9
Compressing the XML Message	6-10
Authenticating the Message	6-10
Ensuring Guaranteed Delivery	6-11
Interpreting the HTTP Status Codes	6-11
Interpreting the XML Reply Message.....	6-12
Sample Reply Message Format.....	6-12
Return Code Values	6-13
Formatting Timestamps.....	6-14
Providing FieldTypes	6-14
Class and Type Values.....	6-14
Subscribing to PeopleSoft.....	6-15
Steps to Subscribe from PeopleSoft	6-15

Creating Subscription Message Components.....	6-16
Configuring the Handler	6-16
Before You Subscribe.....	6-16
Uncompressing Messages	6-16
Interpreting FieldTypes	6-17
Class and Type Values	6-17
XML File Format.....	6-18
XML Field Definition Table	6-19
Application-specific XML Fields	6-24
Common Application Messaging Attributes (PSCAMA).....	6-25
PSCAMA Record Fields	6-26
Developing a Subscription Handler.....	6-28
Simple File Publication Handler	6-28
FileHandlerEntry.class.....	6-29
XSLHandlerEntry.class.....	6-29
SimpleFileHandler.class.....	6-29
Simple XSL Handler	6-29
Administer XSL Handler	6-31
Administer File Handler Add Mode	6-32
Administer XSL Handler Add Mode	6-32
Administer File Handler Edit Mode.....	6-32
Administer XSL Handler Edit Mode	6-32
Simple File Publication Handler Class	6-32
Simple XSL Publication Handler.....	6-35
SimpleFileHandler Java Source Code.....	6-37
Samples.....	6-37
XML Transaction	6-37
Posting XML to an Application Messaging Gateway.....	6-43
Possible Posting Errors	6-43
Third Party to PeopleSoft XML/HTTP Conversation.....	6-44
PeopleSoft to Third Party XML/HTTP Conversation.....	6-46
Ping and Response Sample	6-48
Posting from a C Program.....	6-49

Chapter 7

Application Messaging Architecture

Why Use Application Messaging?	7-2
Terminology	7-2
PeopleSoft Messaging Architecture.....	7-4
Required Components.....	7-4

Nodes.....	7-5
Local.....	7-6
Remote	7-6
Application Server	7-7
Messaging Server Processes	7-7
Brokers and Contractors.....	7-8
Dispatchers and Handlers.....	7-9
Web Server.....	7-10
Sample Message Journey.....	7-11
Publishing to a Local Node	7-11
Publishing to a Remote PeopleSoft Node	7-12
Publishing to a Third Party System.....	7-14

Chapter 8

Messaging Server Administration

Adding Dedicated Message Servers	8-2
Creating and Assigning Dedicated Servers	8-3
Configuring Dedicated servers.....	8-12
Editing Message Server Channel Lists.....	8-12
Removing Unneeded Message Servers	8-15
Configuring Messaging Servers in PSADMIN.....	8-17
Dispatcher Parameters.....	8-17
Handler Parameters	8-18
Configuring PSMBSRV and PSMBHND	8-19

Chapter 9

Administering the Application Messaging Gateway

Overview.....	9-1
The Servlets.....	9-1
Gateway Servlet	9-2
Configuration Servlet	9-2
Reader Servlet	9-2
How it Works	9-2
Working with the Configuration Servlet	9-3
Adding Items to the Handler Directory.....	9-4
Deleting Handlers from the Handler Directory	9-5
Configuring the PeopleSoft Handler.....	9-5
Adding Nodes to the Lookup Table	9-6
Multi-Application Server Nodes	9-8
Editing Nodes.....	9-8

Deleting Nodes.....	9-9
Configuring an External Handler.....	9-9
Configuring a Handler.....	9-10

Chapter 10

Application Message Monitor

Using the Message Monitor Component	10-1
Before You Get Started.....	10-3
Filtering Messaging Information.....	10-3
Maintaining Filter Criteria	10-4
Overview	10-4
Filtering Options	10-5
Status Columns.....	10-5
Message Instances	10-6
Filtering Options	10-7
Output.....	10-7
Pub Contracts	10-8
Filtering Options	10-8
Output.....	10-9
Sub Contracts	10-9
Filtering Options	10-9
Output.....	10-10
Channel Status.....	10-10
Node Status	10-11
Scheduled System Pause Times For Local Node.....	10-12
Testing the Availability of a Remote Node.....	10-14
Queries	10-14
Working with Message Monitor Processes	10-16
Error Notification	10-16
Message Archiving.....	10-18
Using the Message Details Component	10-19
Message Properties.....	10-19
Message Instance Information	10-20
Publication Contracts	10-21
Subscription Contracts	10-21
Message Errors.....	10-22
XML Message Viewer	10-23
Structured Message Viewer	10-23
Using the Structured Message Viewer	10-23
Customizing the Structured Message Viewer	10-24

Purging the Messaging Tables.....	10-26
Printing a Message.....	10-26
Using the Message Monitor Component Interface	10-27

Index

ABOUT THIS PEOPLEBOOK

This book contains information related to the PeopleSoft Application Messaging. Although the Installation and Administration book provides procedures for installing and configuring the components, this book offers information that you'll use over time, not just at installation.

Introduction to Application Messaging is an overview of the message designers used to create messages.

Designing Application Messages is a more detailed step-by-step description of creating messages.

Publishing Messages is a step-by-step description of publishing messages.

Subscribing to Messages is a step-by-step description of subscribing to messages.

Processing Messaging Errors covers how to determine errors during publishing and subscribing to messages.

Third Party Integration gives specific information about how non-PeopleSoft systems can publish to and subscribe from PeopleSoft systems.

Application Messaging Architecture covers the components that make up the application messaging architecture.

Messaging Server Administration presents the menus that are specific to configuring dedicated Messaging Servers.

Administering the Application Messaging Gateway covers the interface you'll use to set up and maintain the Application Messaging Gateway so that you can publish messages to other PeopleSoft Message Nodes as well as third party nodes.

Application Message Monitor covers the panels and procedures associated with monitoring your application messaging system using the Application Message Monitor.

Before You Begin

To benefit fully from the information covered in this book, you need to have a basic understanding of how to use PeopleSoft applications. We recommend that you complete at least one PeopleSoft introductory training course.

You should be familiar with navigating around the system and adding, updating, and deleting information 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.

Related Documentation

To add to your knowledge of PeopleSoft applications and tools, you may want to refer to the documentation of the specific PeopleSoft applications your company uses. You can access additional documentation for this release from PeopleSoft Customer Connection (www.peoplesoft.com). We post updates and other items on Customer Connection, as well. In addition, documentation for this release is available on CD-ROM and in hard copy.



Important! Before upgrading, it is *imperative* that you check PeopleSoft Customer Connection for updates to the upgrade instructions. We continually post updates as we refine the upgrade process.

Documentation on the Internet

You can order printed, bound versions of the complete PeopleSoft documentation delivered on your PeopleBooks CD-ROM. You can order additional copies of the PeopleBooks CDs through the Documentation section of the PeopleSoft Customer Connection Web site: <http://www.peoplesoft.com/>

You'll also find updates to the documentation for this and previous releases on Customer Connection. Through the Documentation section of 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 delivered on your PeopleBooks CD.

Documentation on CD-ROM

Complete documentation for this PeopleTools release is provided in HTML format on the PeopleTools PeopleBooks CD-ROM. The documentation for the PeopleSoft applications you have purchased appears on a separate PeopleBooks CD for the product line.

Hardcopy Documentation

To order printed, bound volumes of the complete PeopleSoft documentation delivered on your PeopleBooks CD-ROM, visit the PeopleSoft Press Web site from the Documentation section of PeopleSoft Customer Connection. The PeopleSoft Press Web site is a joint venture between PeopleSoft and Consolidated Publications Incorporated (CPI), our book print vendor.

We make printed documentation for each major release available shortly after the software is first shipped. Customers and partners can order printed PeopleSoft documentation using any of the following methods:

Internet	From the main PeopleSoft Internet site, go to the Documentation section of Customer Connection. You can find order information under the Ordering PeopleBooks topic. Use a Customer Connection ID, credit card, or purchase order to place your order. PeopleSoft Internet site: http://www.peoplesoft.com/ .
Telephone	Contact Consolidated Publishing Incorporated (CPI) at 800 888 3559 .
Email	Email CPI at callcenter@conpub.com .

Typographical Conventions and Visual Cues

To help you locate and interpret information, we use a number of standard conventions in our online documentation.

Please take a moment to review the following typographical cues:

<code>monospace font</code>	Indicates PeopleCode.
Bold	Indicates field names and other page elements, such as buttons and group box labels, when these elements are documented below the page on which they appear. When we refer to these elements elsewhere in the documentation, we set them in Normal style (not in bold). We also use boldface when we refer to navigational paths, menu names, or process actions (such as Save and Run).
<i>Italics</i>	Indicates a PeopleSoft or other book-length publication. We also use italics for <i>emphasis</i> and to indicate specific field values. When we cite a field value under the page on which it appears, we use this style: <i>field value</i> . We also use italics when we refer to words as words or letters as letters, as in the following: Enter the number 0, not the letter O.
KEY+KEY	Indicates a key combination action. For example, a plus sign (+) between keys means that you must hold down the first key while you press the second key. For ALT+W, hold down the ALT key while you press W.
Jump links	Indicates a jump (also called a link, hyperlink, or hypertext link). Click a jump to move to the jump destination or referenced section.

Cross-references

The phrase For more information indicates where you can find additional documentation on the topic at hand. We include the navigational path to the referenced topic, separated by colons (:). Capitalized titles in *italics* indicate the title of a PeopleBook; capitalized titles in normal font refer to sections and specific topics within the PeopleBook. Cross-references typically begin with a jump link. Here's an example:

For more information, see Documentation on CD-ROM in About These PeopleBooks: Related Documentation.

• Topic list

Contains jump links to all the topics in the section. Note that these correspond to the heading levels you'll find in the Contents window.



Name of Page or
Dialog Box

Opens a pop-up window that contains the named page or dialog box. Click the icon to display the image. Some screen shots may also appear inline (directly in the text).



Text in this yellow bar indicates information that you should pay particular attention to as you work with your PeopleSoft system. If the note is preceded by **Important!**, the note is crucial and includes information that concerns what you need to do for the system to function properly.



Text in this gray bar indicates For more information cross-references to related or additional information.



Text within this bar outlined in red indicates a crucial configuration consideration. Pay very close attention to these warning messages.

Comments and Suggestions

Your comments are important to us. We encourage you to tell us what you like, or what you would like changed about our documentation, PeopleBooks, and other PeopleSoft reference and training materials. Please send your suggestions to:

PeopleTools Product Documentation Manager
PeopleSoft, Inc.

4460 Hacienda Drive
Pleasanton, CA 94588

Or send comments by email to the authors of the PeopleSoft documentation at:

DOC@PEOPLESOFT.COM

While we cannot guarantee to answer every email message, we will pay careful attention to your comments and suggestions. We are always improving our product communications for you.

Introduction to Application Messaging

Overview

Application Messaging, based on the “publish-and-subscribe” model, allows PeopleSoft applications to integrate with each other and with third party applications. This model provides integration that is close to real-time, which means that the publisher need not be connected to the subscriber when publishing the data (similar to how email works). On one end, a message is created and published and on the other end, the message is delivered to any number of subscribers.



For detailed information about key features of application messaging, its architecture, required components, and sample message journey, see *Application Messaging Architecture*.

Message Designers and Objects

Application messages are the fundamental building blocks comprising the application messaging system. Messages are self-describing entities formatted in XML. Each message contains data to be distributed among systems at runtime.

To enable the creation and delivery of application messages, you’ll define the following object types in Application Designer:

Message Objects	What they do...
Message definitions	Stores the information about how a single application message is constructed. Each message definition has a multi-level structure, similar to components, that defines the data to be inserted into the application message at runtime.
Message channels	Channels correspond to groups of message definitions. They help order messages, enhance scalability, and provide a simple way to define processing characteristics of many similar messages as a single group. Channels include message routings, which define the mappings between message nodes on the messaging network.
Message Nodes	The physical systems (application servers or databases) connected to the messaging network.

CHAPTER 2

Designing Application Messages

You use Application Designer to define three types of objects related to application messaging:

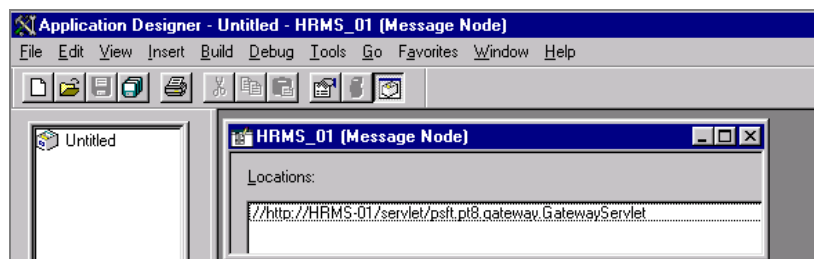
- Message nodes
- Message channels
- Message definitions

If you are creating *new* objects, they must be defined in the order listed above. That is, each message definition must be assigned to a message channel in order to be saved, and each message channel must be linked to a message node. Therefore, if you want to define a new message, ensure that an appropriate *message channel* already exists for the message (either define a new message channel, or verify that an appropriate message channel already exists). Furthermore, before you define a new message channel, make sure that an appropriate *message node* exists for the new message channel.

Defining Message Nodes

A *message node* is an object that represents a publishing or subscribing system on the messaging network. A message node will often relate to an application server or database name.

Use the Message Node Designer to view and edit locations of nodes on the messaging network.



Message Node Designer

To create a new message node

1. Select **File, New** from the menu.

Select *Message Node* from the list of object types.

2. Add locations to the new message node.

To add a location to a message node

1. Select **Insert Location** from the menu.

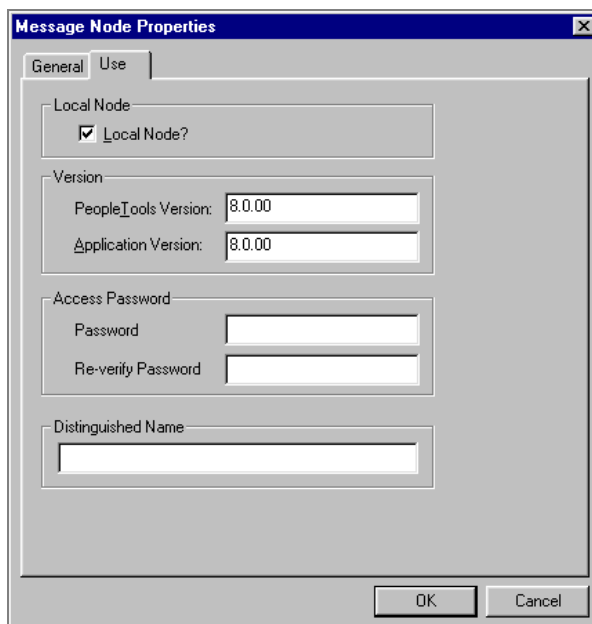
Enter the URL of the node you are defining. The location will be added to the **Locations** list. You can choose to have multiple locations for one node. If more than one location is listed for a node, the system will try contacting the additional URLs if the first one is down or busy.



For more information on setting up nodes, see *Administering the Application Messaging Gateway*.

2. Set the message node properties for the new location.

Select **File, Object Properties** to view and edit the properties of the message node.



The image shows a Windows-style dialog box titled "Message Node Properties". It has two tabs: "General" (selected) and "Use". The "General" tab contains several sections: "Local Node" with a checked checkbox labeled "Local Node?"; "Version" with two text boxes, "PeopleTools Version:" and "Application Version:", both containing the text "8.0.00"; "Access Password" with two text boxes, "Password:" and "Re-verify Password:", both empty; and "Distinguished Name" with a single empty text box. At the bottom right are "OK" and "Cancel" buttons.

Message Node Properties screen

Check the Local Node checkbox if this is the local node on your system. There must be one **Local Node** defined for all the nodes in a single database—the local node represents that database.



The **PeopleTools Version** and **Application Version** are not used at this time, and are for documentation purposes only.

3. Enter the Access Password.

The ***node access password*** verifies the authenticity of incoming messages—in other words, how the local node checks that an incoming message from a remote node actually came from that node.

You are not required to enter a node access password to save the node definition. However, if a user enters a password, then all incoming XML messages must include the node ID (or node name) and this password.

4. Enter the Distinguished Name.



For more information about a *distinguished name* see Digital Certificates.

Defining Message Channels

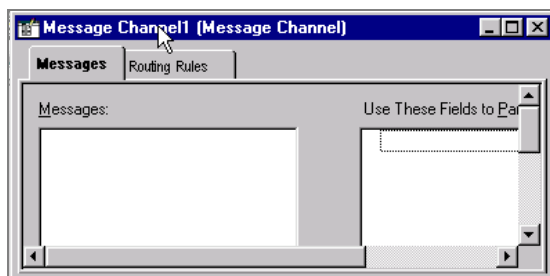
Channels are a logical grouping of messages. Each message must belong to one and only one message channel. Channels are used for the following:

- Ordering of messages
- Routing of messages to specific message nodes
- Increasing throughput when using channel partitioning
- Security
- Easier maintenance of messages with common configuration properties
- Defining the processing attributes for timeout parameters, error thresholds, and delivery type (Guaranteed or Best Effort Delivery)

To define a message channel

1. Open an existing message channel or create a new one.

To create a new message channel, select **File, New**; then select ***Message Channel***.



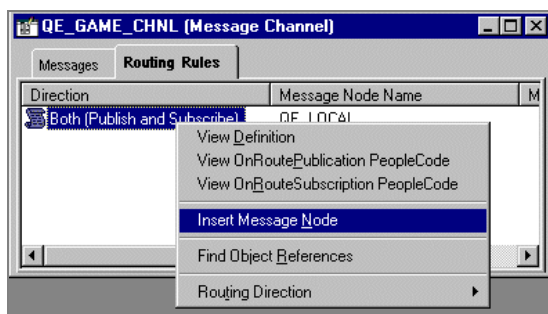
New Message Channel Screen



Note. You must define a channel before you can define messages. When you create a new channel definition, the list of messages is blank. You can name the channel and set the attributes, but you will have to wait until you add messages in the Message Designer before you can partition the channel.

2. Click the **Routing Rules** tab.

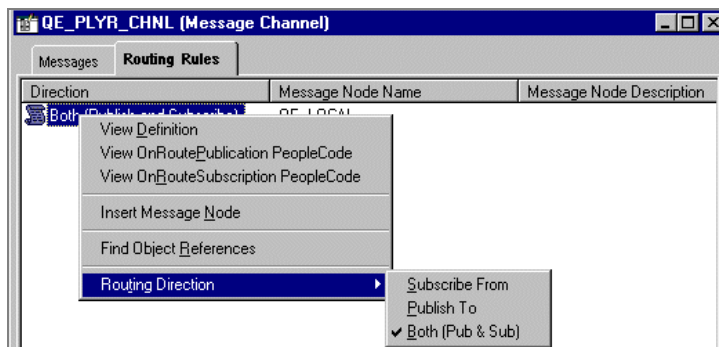
Right-click in the screen to see the pop-up menu. Select **Insert Message Node**. Enter the Message Node name in the Insert Message Node screen in the Name field or use a partial search of existing nodes in Selection Criteria area.



Insert Message Node in Channel Screen

3. Set the Routing Direction.

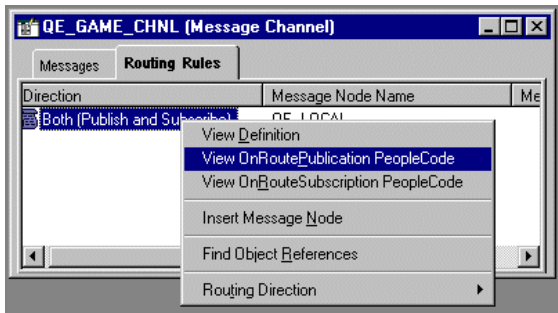
Right-click in the screen to set the routing direction. Choose one: **Subscribe To**, **Publish From**, or **Both**.



Set Routing Direction in Message Designer

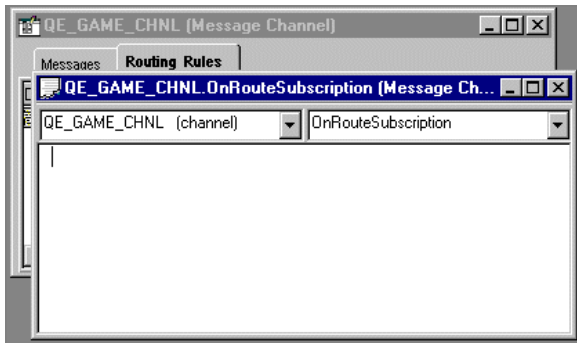
To define message channel routing rules

1. Right-click in the **Routing Rules** tab open window.



Selecting View PeopleCode

2. Select **OnRoutePublication** or **OnRouteSubscription** PeopleCode, depending on your specific message definition.



Enter Publication or Subscription PeopleCode

3. Enter your publication or subscription PeopleCode, as needed.

OnRoutePublication

The OnRoute Publication PeopleCode executes after the message is published. *It determines to which nodes the message will be routed.* This PeopleCode event can be complex or simple. For example, a simple OnRoutePublication PeopleCode function or routine returns a list of subscribing nodes to route the message. A more complex routine could call a function that might check a user-maintained table specifying which nodes receive a particular message.

The following sample of OnRoutePublication PeopleCode routes the messages included in the channel to the PSFT_EP and PSFT_HR nodes

```
Local string &MSGNODENAME;

Local array &NODE_ARRAY;

&NODE_ARRAY = CreateArray();

&MSGNODENAME = "PSFT_EP";

&NODE_ARRAY.Push(&MSGNODENAME);
```

```

&MSGNODENAME = "PSFT_HR";

&NODE_ARRAY.Push (&MSGNODENAME);

ReturnToServer (&NODE_ARRAY);

```

OnRouteSubscription

You can use OnRouteSubscription PeopleCode in addition to the message channel definition to control routing within the subscribing node—that way you only receive those messages that you want. Any OnRouteSubscription PeopleCode function must return a true value if you want the message to be subscribed by the node, or false if you do not.

The following sample of OnRouteSubscription only subscribes to messages that come from PubNode PSFT_EP.

```

Local Message &MSG;

&MSG = GetMessage();

If &MSG.PubNodeName = "PSFT_EP" Then

    ReturnToServer ( True);

Else

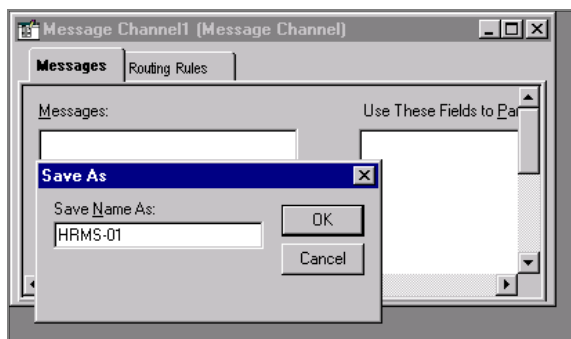
    ReturnToServer ( False)

End-If;

```

4. Save the message channel.

If this is a new channel, you must save it before you can add messages.



Save New Channel Screen

Deleting a Message or Channel in Upgrade

In order to delete a message definition or channel in an application upgrade project, you must *archive* the message in both the source database and the target database. Otherwise, a message displays during the copy process that says the object is in use.

How Channels Are Used at Run Time

Application Messaging guarantees that messages are delivered in the order published and that they are single-threaded on the subscriber. However, message order is not part of the channel definition. Therefore, it is the publisher's responsibility to publish messages in the proper order.

For example, in the channel definition, you cannot specify that a message named PO_CREATE is processed before a message named PO_UPDATE. If the subscriber receives PO_UPDATE before PO_CREATE, it will process PO_UPDATE first.

Also note, if a channel in the publishing node contains different messages and attributes (throughput type, timeout, and so on) than a channel *with the same name* on the subscribing node, the subscribing node will use its own channel to single-thread the messages. To ensure that messages are processed on the subscriber as the publisher intended, the channel definitions on the publisher and subscriber must be kept synchronized—that is, the channel definitions must have the same attributes.

Setting Message Channel Properties

You can set unique properties for every message channel you create.

To set message channel properties

1. With a message channel open in Application Designer, select **File, Object Properties**.

The Message Channel Properties dialog displays.



The Message Channel Properties Dialog

2. Click the **General** tab to view the basic object properties.

The properties on this tab are the standard PeopleTools general properties, and include the object **Description**, **Comments**, and information about the last update to the object.

3. Click the **Use** tab to view and define how this message channel is used at runtime.
4. Define the Message Channel Status.

Run	Messages in this message channel are received and processed normally.
Pause	Messages are received but not processed until the status is reset to Run.

5. Specify whether or not to **Archive Messages**.

Select the **Archive Messages** checkbox to have messages in this message channel archived. If you do not select this check box, the messaging archive process will delete the queue entries that have been processed.

This checkbox also controls whether Archive or Delete action is available in the Message Details component of the Application Message Monitor.

6. Set the **Quality** of service.

Best Effort	The network routes the message to an available node.
--------------------	--

Guaranteed

If the message server fails to deliver the message, it will retry until the **Time-out Period** expires. Then it will mark the message as *Timeout*. Once the subscribing system is ready to receive the message, the system administrator can resubmit any message that timed out.

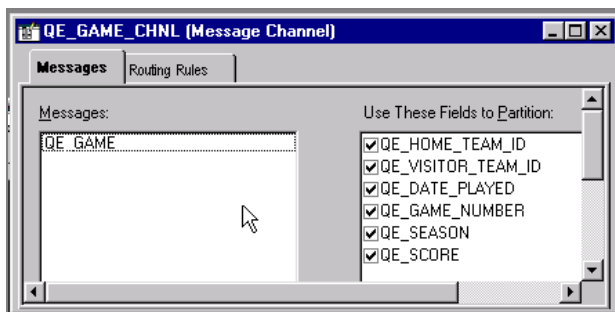
Adding Messages to Message Channel

Before you can add messages to any message channel, the message definition must already exist.

If you have already created your messages or are using previously created messages, open the Message Designer and specify the new channel. When you highlight the Message Channel object, the messages automatically display under the Messages tab.

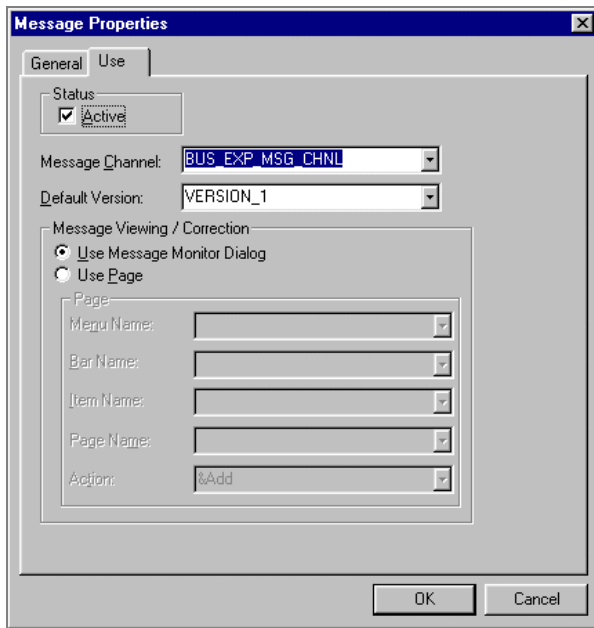
Channel Partitioning

The Channel Designer permits some performance tuning called *partitioning*. You can select one or more of the common fields of the Level 0 records for the messages in that channel for partitioning. By selecting one common field, instead of processing every message in the order that they were published, only the messages with the same value in a partitioned field will be processed sequentially. If the values are different, the message can be processed in parallel.

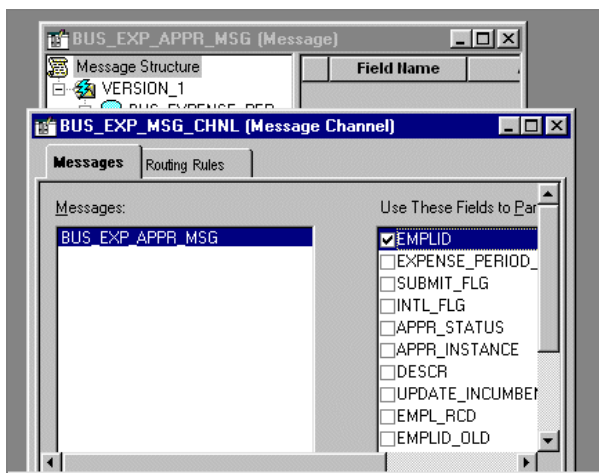


Channel Partitioning Screen

If you do not specify any partition fields, the system processes all instances of messages in the channel sequentially.



Assign Channel in Message Properties



Message in Channel Screen

The left side of the message channel definition displays a list of messages that “belong to” this message channel (have it defined in their object properties). The list of messages is read-only. You cannot insert or delete message definitions directly from the message channel definition, because the channel is a property of the message definition. In order to change which channel a message definition is associated with, you use Message Designer.

To partition the channel

1. Select the checkbox next to the fields in the message are to be partitioned.

The right side of the message channel definition displays a list of fields within this message channel in the **Use These Fields to Partition** box. This list consists of all the level zero fields in common among all messages in the channel.

Defining Application Messages

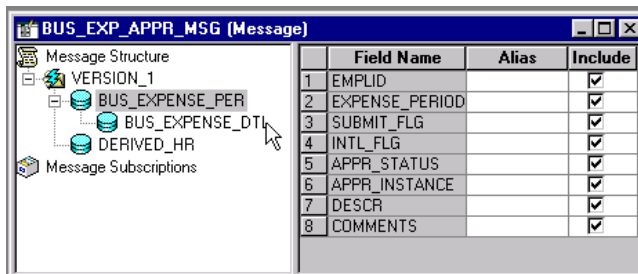
You use Message Designer to create and modify message definitions. At runtime, when a specific message is published, application data in the record is inserted into the message according to the records specified in the message definition.



Note. If you have a message definition that includes character fields defined as uppercase, when the message is subscribed to, character data for those fields is automatically converted to uppercase.

Message Designer Views

Message Designer consists of two main views—the message structure view (on the left) and the list view (on the right).



Message Designer Views

Message Structure View

The message structure view shows several elements of the message definition, including:

- Different versions of the message structure.
- Record definitions (and child records) to be included in each message version.
- A list of subscription processes used to process the message.

List View

The list view displays the fields comprising the selected record in the message structure.

The **Alias** column in the list view enables you to specify an XML field tag value for the message that differs from the original field name. The subscribing system can use this value instead of the original field name from the publishing system. If the publisher does not specify an alias value, the subscriber uses the original field name.



The **Alias** field is useful for cross release and third party integration. The **Include** check box is a security feature that controls whether the field is to be published in the message.

Creating a New Message Version

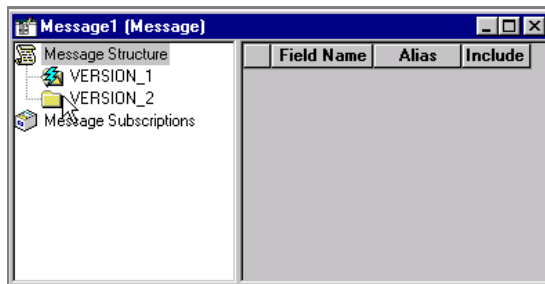
To create a new message version

1. Open an existing message definition or create a new one.

To create a new message, select **File, New**; then select **Message**.

2. With a message definition open, select **Insert Version** from the menu.

You can also right-click the message structure icon and select **Insert Version** from the pop-up menu. Selecting **Insert Version** adds a new version to the Message Structure tree. In the new version, you can insert records or child records. Note that a new version *is not* added to the Message Subscription tree, but you can insert a corresponding message subscription, if desired.



Inserting Versions into a Message Definition

Default Version Names

Each message version has a default name of VERSION_*N*. This naming scheme starts with VERSION_1 and increases in increments of one.

Default Message Structure Versions

The structure version marked with the lightning bolt represents the default message version.

To set the default message version

1. Right-click on the message version and select **Set As Default** from the pop-up menu.

You can also set the default version on the Properties dialog. Select the **Use** tab, and then select the **Default Version** from the list of available versions.

To rename a message version

1. Click the message name once and type over the existing name.

You can also right-click on the message version and select **Rename** from the pop-up menu.

Inserting a Record into a Message Version

To insert a record into a message version

1. In the message structure tree, select the message version into which you want to insert a record.
2. Select Insert, Child Record.

The very first record you insert into a message version must be inserted this way. After a record has been inserted into a message version, you can select **Insert, Record** to insert additional records on the *same level* as the first record. You can also select **Insert, Child Record** to insert records in to the next level of the hierarchy, while an existing record is selected in the message structure tree.



The **Insert Record** dialog remains active until you click **Close**. This allows you to insert multiple records quickly and efficiently, without having to return to the menu for each record to be inserted.

To change the order of records or child records in the message, use the up and down buttons on the toolbar, or drag and drop items to another location in the structure.



Any change to the underlying record definition used in a message definition, automatically results in a corresponding change to the message definition.

Inserting a Child Record

To insert a child record into a message version

1. Select the “parent” record in the message structure tree.

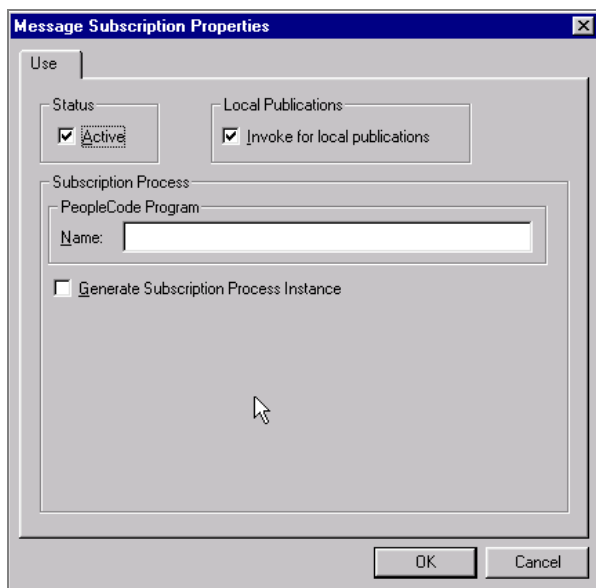
2. Select **Insert, Child Record** from the menu.

Inserting a Message Subscription

To insert a message subscription into a message version

1. With a message definition open, select **Insert, Message Subscription** from the menu.

You can also right-click any part of the message structure tree and select **Insert Message Subscription** from the pop-up menu. The **Message Subscription Properties** dialog displays:



The Message Subscription Properties Dialog

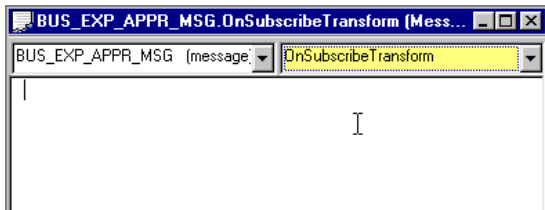
2. Select **Status**.
3. Determine if you want the message to **Invoke for local publications**.

If the **Invoke for local publications** checkbox is selected, the message is published to the local node.

To avoid subscribing to a message just published, de-select (uncheck) the **Invoke for local publications** checkbox.

4. Enter the name of the **PeopleCode Program**.

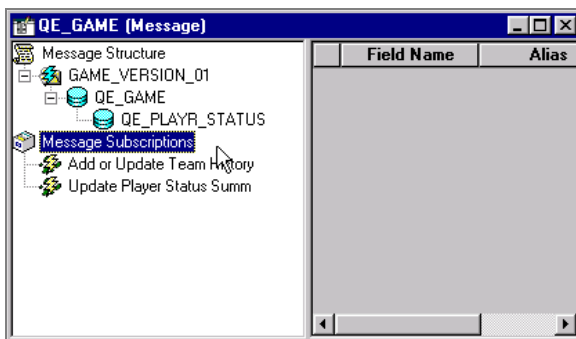
Enter the name of the PeopleCode Program that is to be executed as the Subscription Process. The PeopleCode Editor displays when you click **OK**.



PeopleCode Editor for Subscription Process

5. Enter the subscription PeopleCode.

The subscription process is a PeopleCode program that is saved as part of the message definition. At runtime, this PeopleCode program executes upon receipt of a new message. After you enter the subscription PeopleCode, the new subscription process displays in the message structure view, beneath *Message Subscriptions*. A single message definition can contain multiple subscription definitions.



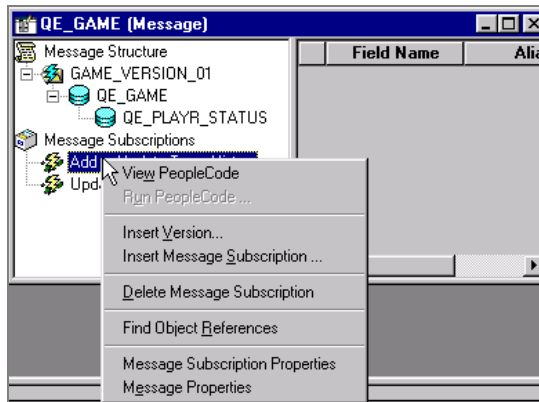
Multiple Message Subscriptions in Message

6. Select Generate Subscription Process Instance, if desired. (?)

View Subscription PeopleCode

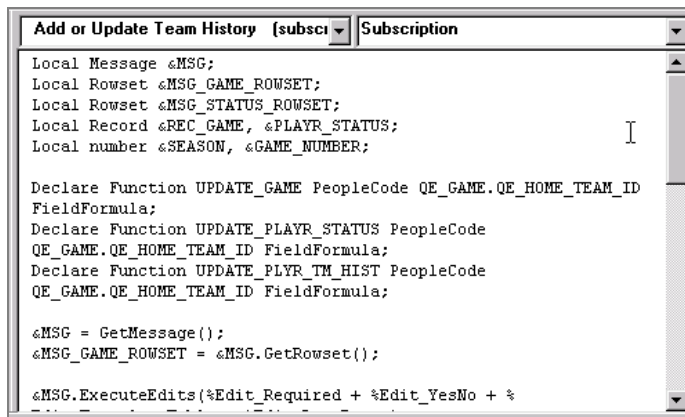
To view subscription PeopleCode

1. Right-click on the **Message Subscriptions** PeopleCode program.



View Subscription PeopleCode

2. Choose View PeopleCode.



Display of Subscription PeopleCode

Example of Single Node Publish and Subscribe

This example shows a simple single node publish and subscribe using SavePostChange PeopleCode in a component.

```
Local Message &MSG;
Local Rowset &COMPONENTROWSET;

&COMPONENTROWSET = GetLevel0();
&MSG = CreateMessage(MESSAGE.WANDA_PERSPUB);
&MSG.CopyRowsetDelta(&COMPONENTROWSET);
&MSG.Publish();

Local Message &MSG;
Local Rowset &LEVEL0;
Local Record &PERSINFO;
```

```

&MSG = GetMessage();
&PERSINFO = CreateRecord(RECORD.WANDA_PERSBC);

&LEVEL0 = &MSG.GetRowset();
&REC = &LEVEL0(1).PERSONAL_DATA;

&REC.CopyFieldsTo(&PERSINFO);
&PERSINFO.Insert();

```

Viewing and Changing Message Properties

To view or change message properties

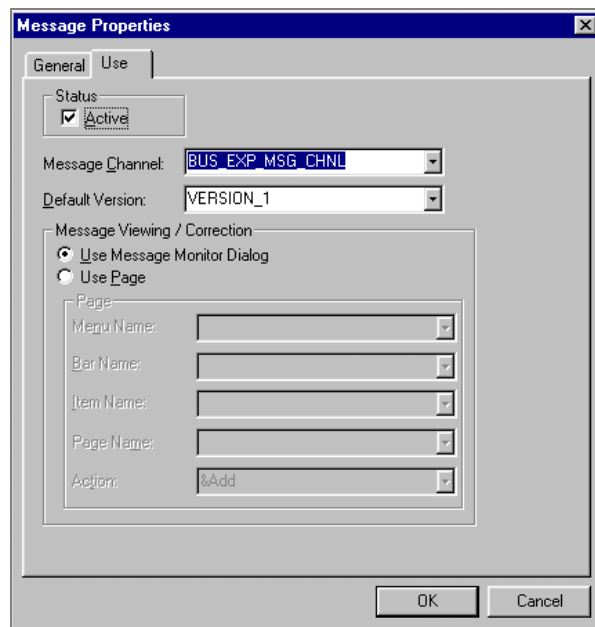
1. Open the message definition.
2. Select File, Object Properties.

Alternatively, you can right-click anywhere inside of the list view and select **Message Properties** from the pop-up menu.

3. Select the **General** tab to view the basic object properties.

The properties on this tab are the standard PeopleTools general properties, and include the object **Description**, **Comments**, and information about the last update to the object.

4. Select the **Use** tab to view and define how this message definition is used at runtime.



The Message Properties Dialog

5. Specify the **Status** of the message.

The **Active** check box determines whether a message can be published. To turn off the publish process for a message, inactivate the message using this check box. You could also delete or comment out the PeopleCode used to publish the message, but turning off the **Active** check box is a simpler alternative.

6. Specify a Message Channel.

You must specify a **Message Channel**. You cannot save a message definition without specifying a Message Channel name.

7. Specify a Default Version.

Select a **Default Version** for this message; that is, the one to be published or subscribed to at runtime. All the message versions for this message are listed here.

8. Specify options for viewing and correcting messages at runtime.

The **Error Viewing/Correction** group box is where you specify how to view and correct this message within the Message Monitor.

Use Message Monitor Dialog	Select this to view a simplified version of the message with the Message Monitor.
Use Page	Select this to view the page to use for correcting.

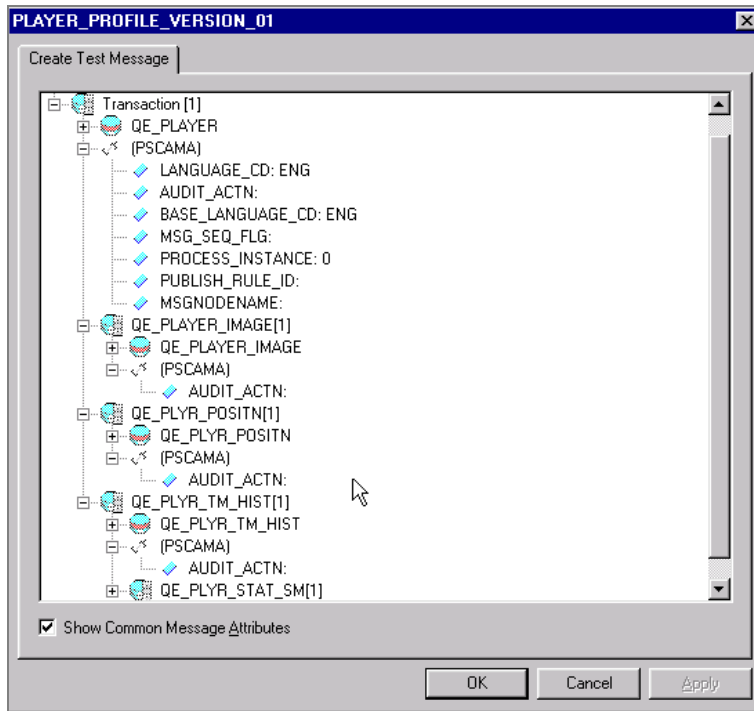
Creating a Test Message

A test message gives you useful information about the selected message definition and its associated subscription process. Because the message is not actually published, creating a test message is not intended to be a *full* publish-and-subscribe test.

To create a test message

1. Select the message version you want to test in the Message Designer.
2. Select **Insert, Create Test Message** from the Message Designer menu.

You can also right-click on the desired message version and select **Create Test Message** from the pop-up menu. A test message appears showing the structure of the message as it would be published. Checking the **Show Common Message Attributes** option displays the PSCAMA attributes in the test message.



Creating a Test Message

3. Enter sample data for the fields in the message.

You can enter sample data into any field shown on the **Create Test Message** tab. To enter sample data for a field, double-click the field and enter a value. You can also right-click on the field, and select **Edit Value**.

4. Click **OK** to “publish” the test message.
5. Use Message Monitor to view the results of the test message.



For more information about the viewing results, see Using the Structured Message Viewer.

Cross-node Messaging

Before you plan to publish and subscribe across nodes, ensure the following:

- The message definition name must be the same in both the publish and the subscribe databases
- The channel definition name must be the same in both the publish and the subscribe databases
- That each database has already been set up with a "Local" node. In addition, the node definition names for the Local Nodes should be unique across all databases (for example, do not have two databases that have their local node named MY_LOCAL_NODE)

- The node definition names must match what is defined in the PeopleSoft Application Messaging Gateway lookup table
- The Application Messaging Gateway is already set up with the publishing and subscribing nodes defined.

Setup for Cross-node Messaging

1. In your publishing node, define the "Remote" Node for the subscribing database, including the location of the Application Messaging Gateway servlet.
2. In your publishing node, define the routing rules in the channel definition in which your message belongs. Route so that the channel can "Publish To" the subscribing node.
3. In your subscribing node, define the "Remote" node for the publishing database, including the location of the Application Messaging Gateway servlet.
4. In your subscribing node, define the routing rules (Message Nodes tab) in the channel definition in which your message belongs. Route so that the channel can "Subscribe From" messages from the publishing node.
5. Log into the publishing node using an application server. Test connectivity to the subscribing node.



For more information about testing connectivity, see Testing the Availability of a Remote Node.

6. Publish your message. Check the status in the publishing node's Node Status in Message Monitor—it should be set to "Done."
7. Check that you see the subscribing node's Message Monitor. The Subscription Contract Status will be set to either "Done" or "Error" (depending on the results of your subscription validation).

CHAPTER 3

Publishing Messages

This section describes the steps needed to publish application messages. Use the following seven basic steps to build and publish an application message:

Step	Task	Use this tool...
1	Design the Message Application	Design tools and relevant documentation
2	Define the Messages Nodes	Message Node Designer in Application Designer
3	Define the Message Channel	Message Channel Designer
4	Define the Messages	Message Designer
5	Add Publish PeopleCode	Record Designer, Component Designer, Application Engine, Message Designer
6	Specify the Channel Routing	Message Channel Designer
7	Test	Message Designer, PeopleCode debugger, Application Server PeopleBook that includes information about Message Monitor and PeopleSoft Application Messaging Gateway.

Design the Message Application

Like most application development, publishing and subscribing to messages requires time for designing and coordinating the integration between the publishing and subscribing systems. They must coordinate the following:

- **Business Process Flow.** What are the various business processes and activities of the publisher and subscriber?
- **Message Definitions.** What data needs to be shared between the publisher and subscriber? How many message definitions need to be created to support the data to be shared?
- **Channels.** In what channels do the message definitions belong? In what order should messages be delivered? What dependencies do they have on other messages?
- **Publishing and Subscribing Applications.** What components or Application Engine programs want to publish and subscribe to these messages? How many subscribers are there,

many to one message, and so on?



For more information about designing Message Applications and Design Patterns, see the *Enterprise Components EIP Guidelines and EC Utilities*.

Define the Message Nodes

Each message node represents a publishing or subscribing system of an application message. You can think of a node as a PeopleSoft database. For example, even if the HRMS and FDM databases were on the same server, they would both be defined as *message nodes*.

In the Application Designer, the Node Definition consists of the URLs of the Web servers that represent the remote databases. A Web server is required to communicate from one message node to another message node; this is also known as an HTTP server.

To define a message node, see Defining Message Nodes.

Ensure that your local node (publishing node) has been defined as **local** and the subscribing nodes are defined as remote.



Note. You must define at least one node as your “local” node (that is, the database in which you are working). Once you start testing across databases, you will need to define the other database(s) as nodes, too. Third party applications are also set up as nodes. *If you are running standalone*, you need to define a *Local* node in order to publish to the application message queue, but you will not need to put in a Location (URL) for that node.

Define the Message Channel

Channels are a logical grouping of messages. Each message must belong to one and only one message channel.

To define your message channel, see Define the Message Channel.



Note. You must define a channel before you can define messages. When you create a new channel definition, the list of messages is blank. You can proceed by naming the channel and setting the attributes, but you will have to wait until you add messages in the Message Designer before you can partition the channel.

Set the Channel Routings

Channel routings specify the routing of messages between nodes. To send a message anywhere, you need to define a Channel and Node relationship for both the publish side and the subscribe side. You are required to specify high level channel routing, that is, the *direction* of the message in the channel.



Note. If you are working in a single node environment, set the routing as *Publish and Subscribe*.

To define high level channel routings, see Set the Channel Routings.

Low-level Channel Routing

Low-level channel routing is content-based routing. For example, you can route certain messages to certain nodes depending on pre-defined circumstances. Use PeopleCode to define any low-level routing—the publisher uses OnRoutePublication PeopleCode and the subscriber uses OnRouteSubscription PeopleCode. Insert the PeopleCode in the Routing Rules tab of Channel Designer.



Note. Routing Rule PeopleCode will have access to the entire message, so it can make routing decisions based on any of the fields in any of the records in the message.

Define the Messages

You create a message and its structure from the same tables (records) in which the data is stored. You can use any records for your message—tables, views, derived work records, and subrecords.

The message definition can have any number of levels, and any combination of levels—that is, you are not limited to Levels 0 –3 as you are in a component; you are also not limited to only having one Level 0 record.

To define a new message, see Define the Messages.

PSCAMA Table

PSCAMA, an acronym for *PeopleSoft Common Application Message Attributes*, is a record that PeopleTools adds to every level of the message structure during processing. It is not accessible in the Message Designer, but you can reference it in the publishing and subscribing PeopleCode and see it in the Message Monitor. PeopleCode processes this record in the same way as any other record.

You can set PSCAMA record fields in Publish PeopleCode in order to give the subscriber basic information about the message, for example the Language Code or Audit Action Code.

Additional tips about PSCAMA include:

- You can update values on this common record just like any other record in the message definition when publishing the message.
- The subscription process can access the fields on this record like any other field in the message.



Note. To view the PSCAMA fields, use the Record Designer in Application Designer.

PSCAMA Record

The following screen shows the PSCAMA record:

PSCAMA (Record)							
Record Fields		Record Type					
Num	Field Name	Type	Len	Format	H	Short Name	Long Name
1	LANGUAGE_CD	Char	3	Upper		Lang Cd	Language Code
2	AUDIT_ACTN	Char	1	Upper		Action	Action
3	BASE_LANGUAGE_CD	Char	3	Upper		Lang Cd	Language Code
4	MSG_SEQ_FLG	Char	1	Upper		Message Seq	Message Sequence
5	PROCESS_INSTANCE	Nbr	10			Instance	Process Instance
6	PUBLISH_RULE_ID	Char	30	Upper		Publish Rule	Publish Rule ID
7	MSGNODENAME	Char	15	Upper		Msg Node	Message Node Name

PSCAMA Record

PSCAMA Fields

Field Name	What it does...
LANGUAGE_CD	Language code in which the message is published. When publishing from components, the system sets this field to the operator's default language code. The application developer can override this if necessary.
AUDIT_ACTN	Audit Action code which identifies the row of data as an Add, Update, or Delete, and so on.
BASE_LANGUAGE_CD	Base language code of the publishing database. Used by the generic full table subscription PeopleCode to help determine which tables to update.
MSG_SEQ_FLG	Indicates whether the message is a header (H), trailer (T) or contains data (blank). The subscribing database can use this field to initiate processes. For

	example, the header message may cause staging tables to be cleared while the trailer would indicate that all the data has been received and a update job should be initiated
PROCESS_INSTANCE	Process Instance of the batch job which created the message. This can be used by the subscribing database along with the publishing node and publication id to uniquely identify a group of messages from the publishing node.
PUBLISH_RULE_ID	Use as an audit field to indicate which Publish Rule was used to create the message, and by routing PeopleCode to locate the chunking rule used. Then the chunking rule determines to which nodes the message gets published.
MSGNODENAME	The node to which the message will be published. This is an optional field that is passed to the Publish utility by the application AE program. Routing PeopleCode must look for a value in this field and return it to the application server.

Field Name: MSG_SEQ_FLG

	Value	Active	Eff Dt	Long Name	Short Name
1	H	<input checked="" type="checkbox"/>	01/01/1900	Header Message	Header Msg
2	T	<input checked="" type="checkbox"/>	01/01/1900	Trailer Message	Trailer

Message Sequence Flag Translate Values

Audit Action Codes

Rows in the message have an audit action field to denote what the transaction was. The audit action is required so that the subscribing system knows how to process the incoming data. The following table lists the valid audit action codes and a corresponding description.

Cod e	Description
A	Add row (insert)
D	Delete row
C	Change row (updated, but no key fields changed).
K	Change row (updated, and at least one key field changed). This is the old value. ¹
N	Change row (updated, and at least one key field changed). This is the new value. ¹
""	Blank means that nothing in the row has changed. This is the default value, and also the value used to tag the parent rows of children that have changed.

¹The audit codes A, C, and D are set by the system when a record is added, changed, or deleted, respectively. There are some cases, for example in effective dated records, where the user may change Key Field value—that is, Effective Date. In response to such a user action, the system creates two records: one with an audit code of N, and another with the audit code of K. The N-code record contains all the new values, while the K-code record retains the old values.

Field Name: AUDIT_ACTN

	Value	Active	Eff Dt	Long Name	Short Name
1	A	<input checked="" type="checkbox"/>	01/01/1900	Add	Add
2	C	<input checked="" type="checkbox"/>	01/01/1900	Change	Change
3	D	<input checked="" type="checkbox"/>	01/01/1900	Delete	Delete
4	K	<input checked="" type="checkbox"/>	01/01/1900	Key Change Old Key	Old Key
5	N	<input checked="" type="checkbox"/>	01/01/1900	Key Change New Key	New Key

Audit Action Translate Values

Checking the Audit Action

The following shows subscription PeopleCode that checks the PSCAMA to determine what the Audit Action code is. It also checks the Language code to determine if 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;

/*****
/***** Update 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;

```

Add Publish PeopleCode

You can publish application messages using PeopleCode functions and methods from the following objects:

- Components
- Application Engine programs
- Component interfaces
- Subscription programs

<i>If you publish a message in a ...</i>	<i>Use...</i>
Component	SavePostChange PeopleCode on a record or component
Application Engine program	PeopleCode in Application Engine step
Component interface	PeopleCode on a record or component invoked by the Save method
Subscription program	Subscription PeopleCode

For more information about publish and subscribe PeopleCode functions, see Using Methods and Built-in Functions in the PeopleCode PeopleBook.

Where to Put Your PeopleCode

- **Record.** Put your PeopleCode in the **SavePostChange** event of the record if you want the publish code to fire every time the record is accessed.
- **Component.** Put your PeopleCode in the **SavePostChange** event of the component if you want the publish code to fire only for that specific component.
- **Level 0 Record.** Put your PeopleCode on the Level 0 record by default—unless you are publishing changes only to child records, then use Level 1.

PeopleCode Statements to Publish

In order to publish a message, you need to do the following in PeopleCode:

- Create an instance of the message object
- Copy the data into the message object
- Publish the message to the message queue



For more information about PeopleCode, see Understanding PeopleCode and Events.

Sample Publish PeopleCode

This is a sample of adding PeopleCode to the Personal Data component in HRMS_01; the Component is PERSONAL_DATA2_GBL and PeopleCode event is SavePostChange:

```
Local Message &MSG

Local Rowset &ComponentRowset;

/*Get a pointer to the pagebuffer rowset */

&ComponentRowset = GetLevel0();

/*Create an instance of the PERS_DATA_INCR msg object */

&MSG = CreateMessage(MESSAGE.PERS_DATA_INCR);
```

```

/*Copy the changed rows from rowset to msg object */

&MSG.CopyRowsetDelta(&PageGroupRowset);

/*Publish the message */

&MSG.Publish();

```

Tips on Publish PeopleCode

- Use `SelectByKey()` whenever possible to get data that is not in the page buffer.
- *Shortcut:* If your record names are the same, use the standard Record methods, `SelectByKey`, `Insert`, `Update`, and so on, on the Message Records.
- Use `If&Msg.Size > %MaxMessageSize` at Level 0 to control message size when dealing with multiple transactions.
- Note the following differences between Component and Application Engine Publish:
 - Components uses a page processor to retrieve data and execute commit logic; while Application Engine does not.
 - Components can have only one Level 0 row. Application Engine allows multiple Level 0 rows.

Test

As with other types of software development, make sure that you adequately unit test and system test your application messages.

- **To unit test a published message.** You can easily unit test your application message just by triggering the PeopleCode that publishes the message, and then viewing it in the Message Monitor. From the Message Monitor, you can view the contents of each field to verify that the message data is formatted correctly. You can also trigger a message with the Test Message feature in the Message Designer.
- **Run PeopleCode debugger.** Execute the following PeopleCode with the PeopleCode debugger to see the `&MSG` object.

```

Local Message &MSG

Local Rowset &RS;

```



```

/*Get a pointer to the page buffer rowset */

&RS = GetLevel0();

/*Create an instance of the msg object */

&MSG = CreateMessage(MESSAGE.MY_MSG);

/*Copy the changed rows from rowset to msg object */

&MSG.CopyRowsetDelta(&RS);

/*Publish the message */

&MSG.Publish();

```

Write a simple subscription process

You can load the message data to a temporary table to by writing a simple subscription process, as follows:

```

Local Message &MSG;

Local Rowset &MSG_RS;

Local Record &PERS_DATA_TEMP;

&MSG = GetMessage();

&MSG_LEVEL0 = &MSG.GetRowset();

&PERS_DATA_TEMP = CreateRecord(RECORD.PERS_DATA_TEMP);

For &I = 1 to (&MSG_LEVEL_0.RowCount)

    &MSG_LEVEL_0(&I).PERSONAL_DATA.CopyFieldsTo(&PERS_DATA_TEMP);

    &PERS_DATA_TEMP.Insert();

End-for;

```


CHAPTER 4

Subscribing to Messages

This section describes the steps you must perform to subscribe to application messages. Use the following eight basic steps to subscribe to an application message:

Step	Task	Use this tool...
1	Design the Message Application	Paper and pencil, preferred design tools
2	Define the Messages Nodes	Message Node Designer in Application Designer
3	Define the Message Channel	Message Channel Designer
4	Specify the Channel Routing	Message Channel Designer
5	Define the Messages	Message Designer
6	Add Subscribe PeopleCode	Message Designer
7	Test	Message Designer, Message Monitor, PeopleCode debugger

Design the Message Application

See Design the Message Application.

Define the Message Nodes

Each message node represents a publishing or subscribing system of an application message. You can think of a node as a PeopleSoft database. For example, even if the HRMS and ERP databases were on the same server, they would both be defined as *message nodes*.

In the Application Designer, the Node Definition consists of the URL Locations of the Web Servers that represent the remote databases. A web server is required to communicate from one message node to another message node, this is also known as an HTTP server.

To define a message node, see Defining Message Nodes.

Ensure that you have selected a *local* node (the subscribing node) and you have defined the publishing nodes as *remote*.

Define the Message Channel

Channels are a logical grouping of messages. Each message must belong to one and only one Message Channel.

To define the message channel and specify routing, see [Defining Message Channels](#).



Note. You must define a channel before you can define messages. When you create a new channel definition, the list of messages is blank. You can name the channel and set the attributes, but you will have to wait until you add messages in the Message Designer before you can partition the channel.



Note. Routing Rule PeopleCode will have access to the entire message, so it can make routing decisions based on any of the fields in any of the records in the message.

Define the Messages

You must define the messages to which you are subscribing. You create a message and its structure from the same tables (records) in which the data is stored. You can use any records for your message—tables, views, derived work records, and subrecords.

To define a new message, see [Defining Application Messages](#). You will want to know about the PSCAMA record that is appended to each record in your subscribed message.

Add Subscribe PeopleCode

You subscribe to application messages using PeopleCode functions and methods described in this section.

To add subscription PeopleCode to a message, see [Inserting a Message Subscription](#).

For more information about publish and subscribe PeopleCode functions, see [Understanding PeopleCode and Events](#).

Sample Subscription PeopleCode

Sample 1

This is a sample of adding subscription PeopleCode to a message named PERSONAL_DATA.

```
Local Message &MSG
```

```

Local Rowset &MSG_LEVEL_0, &MSG_LEVEL_1;

Local Record &PERS_DATA, &PERS_PHONE;

/*Get message rowset to process */

&MSG = GetMessage();

&MSG_LEVEL_0 = &MSG.GetRowset();

/*Insert Level 0 rows from message */

&PERS_DATA = &MSG_LEVEL_0.GetRow(1).GetRecord(RECORD.PERSONAL_DATA);

&PERS_DATA.Insert();

.....

```

Sample 2

The following example is 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 Application Messages that may contain
multiple Header (MSR_HDR_INV) transactions that may have multiple line
(DEMAND_INF_INV) transactions. The data is inserted into a 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;

```

Checking Maximum Message Size

There is 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. It can not be set for each individual message definition, but rather for all messages.

PeopleCode that populates a message object should include code similar to the following sample to check the message size before inserting a new Level 0 row.



Note: Always code your %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 application 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 it's 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;
```

Test

As the subscriber, you can use the Message Designer's test message feature to create message instances. See [Creating a Test Message](#).

CHAPTER 5

Processing Messaging Errors

Several kinds of errors can occur during the publication or subscription process. You can monitor these errors and, in many cases, fix them in the Message Monitor.

If you want the application messaging system to handle the error tracking and correction, then you must use the `Exit(1)` function to log the errors, perform a rollback, and stop processing.

If you want to handle error processing in your application tables, then use the `Exit()` function, or just let the subscription process complete normally. This marks the message subscription contract as successful and commits.

Validating Data

An application messaging subscription process can validate incoming data in the following ways:

- Use the **ExecuteEdits** method on the message to invoke the definitional edits.
- In cases where PeopleCode validation functions already exist, for example in a FUNCLIB, or validation logic can be encapsulated within a small set of functions. You can call these functions from within the subscription PeopleCode.
- For complex validation logic, you can call a Component Interface or Application Engine program from your subscription process. This allows you to re-use logic embedded in the component.

The **ExecuteEdits** method uses the definitional edits to validate the message. The following system variables can be specified alone or in combination. Not specifying any variable with the **ExecuteEdits** method executes all the edits.

- `%Edit_DateRange`
- `%Edit_PromptTable`
- `%Edit_Required`
- `%Edit_TranslateTable`
- `%Edit_YesNo`

The following example executes all the edits for all levels of data in the message structure:

```
&MYMSG.ExecuteEdits();
```

The following example executes the Required Field and Prompt Table edits:

```
&RECPURCHASEORDER.ExecuteEdits(%Edit_Required + %Edit_PromptTable);
```

ExecuteEdits uses *set processing* to do the validation. Validation using a component interface or a PeopleCode function is usually done with row by row processing. If a message consists of 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.

The ExecuteEdits methods sets several properties on several objects if there are any errors.

- **IsEditError** is set on the Message, Rowset, Row, and Record object if there are any fields that contain errors
- **EditError**, **MessageNumber** and **MessageSetNumber** are set on the field object that contains the error

If you do not want to use ExecuteEdits, you can set your own errors using the field properties. Setting the **EditError** property to True automatically sets the message property **IsEditError** to True. You can also specify your own message number, message set number, and so on, for the field. When you finish setting the fields that are in error and if you used the **Exit(1)** built-in function, the message status changes to **Error**.

Exit Function

Use the **Exit** function to invoke an application messaging error process when the application finds an error. This same error processing is invoked automatically if PeopleTools finds an unexpected error, for example an SQL error. The Exit function has an optional parameter which affects how the error is handled, shown in the form **Exit()**.

Exit ()

In the Exit () form, everything is committed and the message is marked as complete. You can use this to indicate that everything processed correctly. However, sometimes you may want to use this function to stop program processing. Note, though, that the message status will be set to complete; therefore you cannot detect or access errors from the Application Message Monitor. If errors did occur, the subscription code should write them to a staging table, and then you would have to handle all of the error processing.

The Exit function does the following:

- Sets the message status in the application message queue for the Subscription to Done
- Commits
- Stops execution

Sample Exit () PeopleCode

```
&Msg.ExecuteEdits();

If &Msg.IsEditError then

    App_Specific_Error_Processing();

    Exit();

Else

    Specific_Message_Processing();

End-if;
```

Exit (1)

In this form, the Exit function does the following:

- Executes a rollback
- Sets the message status in the application message queue for the subscription to **error**
- Writes the errors to the subscription contract error table
- Stops execution
- You can view all errors that have occurred for this message in the Message Monitor, even those errors detected by ExecuteEdits.

Sample Exit (1) PeopleCode

```
&Msg.ExecuteEdits();

If &Msg.IsEditError then

    Exit(1);

Else

    Process_Message();

End-if;
```

Correcting Errors

If you anticipate having only a few errors to correct, you can do so in Message Errors of the Message Monitor.

If you have many errors, you may want to create a page that a user could use to correct the data for a particular message. Generally, your user will be correcting errors that were detected when running **ExecuteEdits** or another function as part of your subscription.



Note. If you publish and subscribe to the same message locally, on the same node, remember, only **one** copy of the message exists. Both the publication contracts table and the subscription contracts table point to the same data. If you go into the subscription contracts and edit a message, you are **also** editing the message under the publication contract. The status of **both** messages will be changed to EDITED.

For the search record for your page, you may want to create a view based on the MSGNAME and SUBCONSTATUS fields of the PSAPMSGSUBCON. The MSGNAME will list all the messages that have subscription contracts. The SUBCONSTATUS field has the following values:

Status	Value
ERROR	0
NEW	1
STARTED	2
WORKING	3
DONE	4
RETRY	5
TIMEOUT	6
EDITED	7
CANCELLED	8

If you create a page, you will have to define it as part of your message definition. After you define it, you can either use it as a regular page, or from the Message Details link in the Message Monitor.

Correcting Message Errors in PeopleCode

Sample 1

The following code is located in the PreBuild event of a page that is used to fix message errors. It gets the subscription message as specified by the search record (PSAPMSGSUBCON) and loads the fields of a work record (MS_EDIT_WRK) with the data from the message:

```
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();

MSG_EDIT_WRK.QE_PLAYER_ID = &RS_PLAYER(1).QE_PLAYER.QE_PLAYER_ID.Value;

MSG_EDIT_WRK.QE_PLAYER_STATUS = &RS_PLAYER(1).QE_PLAYER.QE_PLAYER_STATUS.Value;

MSG_EDIT_WRK.DESCRLONG = &RS_PLAYER(1).QE_PLAYER.DESCRLONG.Value;

```

Sample 2

The following code is located in the SavePostChg event. It takes the values from the work record and repopulates the rowset. As 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 message 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 Message Errors in the Message Monitor

See XML Message Viewer or Structured Message Viewer for correcting message errors.

Correcting Data Errors in a Page

If you want to use a pre-defined page to correct data errors, you specify that in message properties.

To correct data subscription contract errors in a page

1. In Application Designer, open your message definition or define a new one.
2. Go to the **Message Properties** screen, **Use** tab.
3. Click **Use Page**, and specify page characteristics.

Message Properties dialog displays:

The screenshot shows the 'Message Properties' dialog box with the 'Use' tab selected. The 'Status' section has the 'Active' checkbox checked. The 'Message Channel' is set to 'QE_PLYR_CHNL' and the 'Default Version' is 'PLAYER_PROFILE_VERSION_01'. In the 'Message Viewing / Correction' section, the 'Use Page' radio button is selected. Below this, the 'Page' section contains several dropdown menus: 'Menu Name' (QE_APPLICATION_MESSAGING_), 'Bar Name' (USE), 'Item Name' (GAME MAINTENANCE), 'Page Name' (QE_GAME_PNL), and 'Action' (Update/&Display All). The 'OK' and 'Cancel' buttons are at the bottom right.

Specify Page for Error Correction

Enter the menu name, bar name, item name, page name, and action from the dropdown lists.

To launch the page

1. Open the Message Monitor.
Select the Sub Contracts page and highlight the message in error.
2. Click Message Details.

View the message details.

CHAPTER 6

Third Party Integration

Application messaging supports the integration of PeopleSoft applications and third party systems by publishing business events as XML-formatted messages.

To *publish* data, third party applications can post XML messages directly to PeopleSoft's Application Messaging Gateway servlet. To *subscribe* to data, third party applications can accept and process XML messages posted by PeopleSoft by adding a custom Java subscription handler (or plug-in) to the Application Messaging Gateway servlet. A *servlet* is a platform-independent 100% Pure Java™ module extending the capabilities of the server.



For more information about servlets and the Java Servlet API, see The Servlets or visit the <http://java.sun.com/products/java-server/servlets/> Web site.

Application Messaging Gateway

The Application Messaging Gateway resides on the Web server and directs messages to their intended destinations.



For more information about PeopleSoft's Application Messaging Gateway, see Administering the Application Messaging Gateway.

Application Messaging SDK

The PeopleSoft Integration Software Development Kit (SDK) installed with your application includes several resources that can assist you in developing and testing application message-based integration between PeopleSoft and third party applications. The SDK is installed to the PeopleSoft home directory (PS_HOME), under **sdk**.



For more information and an overview of the PeopleSoft Integration SDK, see Integration Software Development Kit.

SDK Resources

The SDK includes a message posting utility, source code for the utility and a file handler, and sample messages, listed in the following table. The file locations are relative to **<PS_HOME>\sdk\psappmsg**.

Resource	Files	Description
XML Poster	bin\winx86\psxmlpost.exe	DOS command line executable for publishing an application message to a PeopleSoft node
	classes\psxmlpost.jar	Java executable for publishing an application message to a PeopleSoft node
	src\C++\samples\psxmlpost.cpp	C++ source code for psxmlpost.exe
	src\java\samples\psxmlpost\psxmlpost.java	Java source code for psxmlpost.jar
File handler	src\java\samples\simple_file_handler*.java classes*.class	Java source code and classes for the Simple File Publication Handler supplied with your application
Sample messages	src\xml\samples\psxmlpost_msg\sd_bus_exp.xml src\xml\samples\psxmlpost_msg\sd_bus_exp_formatted.xml	Uncompressed application messages with no header
	src\xml\samples\nopsxmlpost_uncomp_msg\sd_nopsxmlpost_uncomp_msg.xml	Uncompressed application message with PeopleSoft header
	src\xml\samples\nopsxmlpost_comp_msg\sd_nopsxmlpost_comp_msg.xml	Compressed application message with PeopleSoft header
	src\xml\samples\replydoc\sd_replay_doc.xml	Example of the status message sent by the PeopleSoft application to confirm receipt of an incoming message.

Publishing to PeopleSoft

Publishing data to PeopleSoft requires that you be familiar with XML format and some parts of the PeopleSoft Application Messaging system. The following table lists a few basic requirements:

What you need...	Where to find it...
Ability to extract data from the publishing third party system, format it into XML, and perform an HTTP post.	Various Web sites that explain XML and post.

<i>What you need...</i>	<i>Where to find it...</i>
Knowledge of XML fields required for integration.	XML File Format
Understanding of data elements of the PeopleSoft Application Message Definition.	Application-specific XML Fields and the “EIP Catalog” section of the <i>Enterprise Integration PeopleBook</i> .
A connection to a PeopleSoft Application Messaging Gateway.	A PeopleSoft web server, provided with your PeopleSoft application — see Administering the Application Messaging Gateway.
A utility to publish application messages.	PeopleSoft XML Poster , included as an executable and as source code in the PeopleSoft Integration SDK installed with your application.
Support of a PeopleSoft test environment.	PeopleSoft Alliances, http://www.peoplesoft.com/

Steps to Publish to PeopleSoft

There are three basic steps for publishing data to a PeopleSoft system:

- Creating subscription message components in PeopleSoft
- Modifying the PeopleSoft gateway handler
- Creating your message

Creating Subscription Message Components

Using PeopleSoft’s Application Designer, do the following or verify that it has been done:

- Identify the local node of the system to which you are publishing (destination node)
- Define the remote node from which the message is being published (source node)
- Define the message channel routing rules
- Define the message definition
- Write the subscription PeopleCode to process the publication

Modifying the Handler

You must add the destination node, that is, the subscribing node to the PeopleSoft handler. *See Adding Items to the Handler Directory.*

Creating Your Message

You need to perform the following for your message:

- Choose a programming language having the TCP support
- Create the XML data to be posted
- Create an XML file and add the XML data
- Post the file. For example, you can use the `HttpOpenRequest/HttpSendRequest`, a Microsoft Win32® Internet (WinInet) function.
- Analyze the reply message sent by PeopleSoft to confirm success or failure



For more information about external handlers, see [Configuring an External Handler](#).

PeopleSoft XML Poster

The integration SDK includes the PeopleSoft XML Poster, a utility you can use to publish properly formatted application messages to a defined PeopleSoft node. The XML Poster is available in a DOS command line version (`PSXmlPost.exe`) and a Java version (`PSXmlPost.jar`). The messages you publish with the XML Poster don't need to include the PeopleSoft application message header—the program automatically generates an appropriate header based on the parameters you enter.



For more information about the PeopleSoft XML Poster and sample messages provided with the integration SDK, see [SDK Resources](#).

PSXmlPost.exe

`PSXmlPost.exe` is a DOS command line executable. You can provide your application message header attributes as command line parameters, or by interacting with a command line menu.

Location

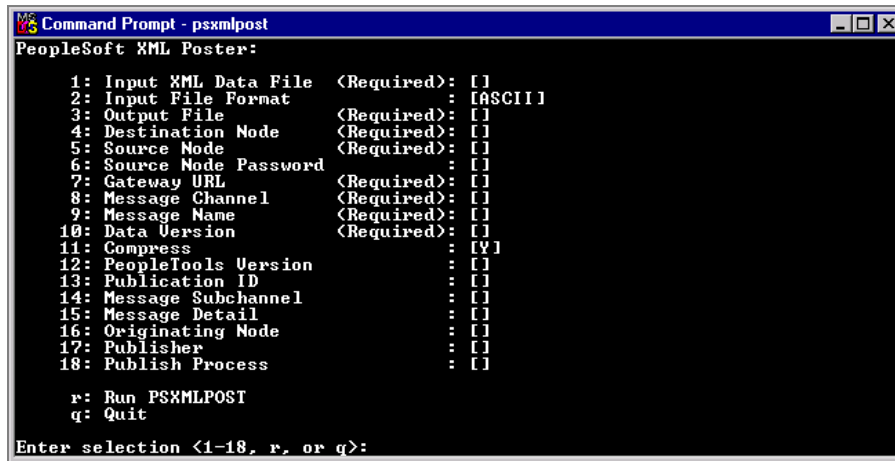
`<PS_HOME>\sdk\psappmsg\bin\winx86`

Using PSXmlPost.exe

To use the command line menu:

1. Launch `PSXmlPost` in a DOS window with no options, or double-click the executable in Windows Explorer.

A command line menu will appear, with the required parameters noted.



```

Command Prompt - psxmlpost
PeopleSoft XML Poster:

1: Input XML Data File  <Required>: []
2: Input File Format    : [ASCII]
3: Output File         <Required>: []
4: Destination Node    <Required>: []
5: Source Node         <Required>: []
6: Source Node Password <Required>: []
7: Gateway URL         <Required>: []
8: Message Channel     <Required>: []
9: Message Name        <Required>: []
10: Data Version       <Required>: []
11: Compress           : [Y]
12: PeopleTools Version : []
13: Publication ID      : []
14: Message Subchannel : []
15: Message Detail      : []
16: Originating Node   : []
17: Publisher          : []
18: Publish Process    : []

r: Run PSXMLPOST
q: Quit

Enter selection <1-18, r, or q>:
  
```

PeopleSoft XML Poster command line menu

2. Enter the line number for each parameter you want to provide or change.
You'll be prompted for an appropriate value.
3. When you've provided all required parameters, enter **r** to publish your message, or **q** to cancel the operation.

If you try to publish the message without valid values for all required parameters, the menu will reappear with the existing entries still filled in.



For more information about the required and optional application message parameters, see PSXmlPost Parameters.

To enter the parameters as command line options:

1. Include each parameter followed by its value with the DOS command:

```
psxmlpost -parameter1 value1 [-parameter2 value2] ...
```

If you've entered all required parameters, the XML Poster will publish your message. If not, you'll see an error message explaining the required and optional parameters.



For more information about the required and optional application message parameters, see PSXmlPost Parameters.

PSXmlPost.jar

PSXmlPost.jar is a Java program. You can provide your application message header attributes as command line parameters, or by interacting with a graphical interface.

Location

<PS_HOME>\sdk\psappmsg\classes

Using PSXmlPost.jar

To use the graphical interface:

1. Ensure that you have Java 2 (JDK 1.2.2) installed.
2. In a DOS window, launch the XML Poster with the following syntax:

```
java -classpath "[path]\psxmlpost.jar" psft.pt8.sdk.PSXmlPost
```

Where *path* is the relative location of PSXmlPost.jar. The graphical interface will appear.

PeopleSoft XML Poster Java graphical interface

3. Enter appropriate values for all the required parameters, and any optional parameters you want.

Required parameters are preceded by an asterisk (*).

4. Click **Post** to publish your message, **Reset** to clear the values from all fields, or **Exit** to quit the program.

If you try to publish the message without valid values for all required parameters, the Response window will list the missing parameters.



For more information about the required and optional application message parameters, see PSXmlPost Parameters.

To enter the parameters as command line options:

1. Ensure that you have Java 2 (JDK 1.2.2) installed.
2. In a DOS window, change to the location of PSXmlPost.jar.
3. Launch the XML Poster with the following syntax:

```
java -classpath "[path]\psxmlpost.jar" psft.pt8.sdk.PSXmlPost [-parameter1 value1] ...
```

If you've entered all required parameters, the XML Poster will publish your message. If not, you'll see an error message explaining the required and optional parameters.



For more information about the required and optional application message parameters, see PSXmlPost Parameters.

PSXmlPost Parameters

The command line parameters for PSXmlPost.exe and PSXmlPost.jar are identical. The following table lists the parameters as they appear in each interface.

<i>DOS Menu</i>	<i>Java GUI</i>	<i>Command Line</i>	<i>Note</i>
1: Input XML Data File	*Input File Name	-infile	required
2: Input File Format	*Input File Format	[-infilefmt]	default = ASCII
3: Output File	(C:\temp\psxmlpost.log is automatically created)	-outfile	required
4: Destination Node	*Destination Node	-to	required
5: Source Node	*Source Node	-from [:password]	required

DOS Menu	Java GUI	Command Line	Note
6: Source Node Password	Source Node Password		[optional]
7: Gateway URL	*URL	-url	required
8: Message Channel	*Channel	-channel	required
9: Message Name	*Message Name	-msgname	required
10: Data Version	*Data Version	-dataver	required
11: Compress	Compress Data	[-compress]	default = Y (compression)
12: PeopleTools Version	PeopleTools Version	[-PSVersion]	[optional]
13: Publication ID	Publication ID	[-pubid]	[optional]
14: Message Subchannel	Subchannel	[-subchannel]	[optional]
15: Message Detail	Message Detail	[-msgdetail]	[optional]
16: Originating Node	Originating Node	[-orgnode]	[optional]
17: Publisher	Publisher	[-publisher]	publisher id [optional]
18: Publish Process	Publish Process	[-pubproc]	[optional]



For more information about application message parameters, see XML File Format.

Before You Publish

This section describes various tasks you might need to perform before you actually publish messages to PeopleSoft. The tasks are:

- Using Publication IDs
- Converting the XML File to UTF-8
- Authenticating the Message
- Ensuring Guaranteed Delivery
- Interpreting the HTTP Status Codes
- Interpreting the XML Reply Message

- Formatting Timestamps
- Providing FieldTypes
- XML File Format

Using Publication IDs

This section describes some general behavior of the publication system and publication IDs.

- If you specify the *publishing node name* and *publication ID* in your XML file, the system uses them as specified. The third party program is responsible for generating unique publication IDs and storing them for tracking purposes.
- If you specify the *publishing node name*, but omit the *publication ID*, the system generates an ID and sets it to the next available publication ID on that channel within the subscribing PeopleSoft database. We provide this as a convenience for those who do not want to manage the generation and tracking of publication IDs. Note the following limitations and consequences:
 - The sequence of publication IDs for local publications will have gaps. This is because the same counter is used for both remote and local publication IDs, and is based on the channel name.
 - If the third party system posts the same message to two different PeopleSoft systems, the publication IDs for those posts will not be the same, even though the message is the same. Furthermore, a given publication key on those two systems may not refer to the same third party publication, making it harder to track messages.
 - PeopleSoft imposes a “one-hop limitation”; that is, publications from a remote node are not forwarded to other remote nodes. Consequently, this case of duplicate publication IDs will not cause a collision.

Converting the XML File to UTF-8

When the message is received by the PeopleSoft database, the XML data is converted to UTF-8 to prevent any UCS2 byte order issues. It is also compressed using the deflate algorithm prior to storage in the database.

Only the contents of the <data> element of the message will be compressed. The output of the deflate algorithm is binary, and therefore not suitable for embedding in an XML document. Because of this, the data is base64-encoded after compression.



For more information about base64 encoding, see Options section of Configuring a Handler; visit the <http://www.oreilly.com/catalog/javacrypt/> Web site; and see examples stored at the <ftp://ftp.ora.com/pub/examples/java/crypto/files/oreilly/jonathan/util/> Web site.

The XML POST document must be converted to UTF-8 before being posted. The HTTP post data is a UTF-8 XML document and the expected response is a UTF-8 XML document.

Sample UTF-8 XML

```
content-type: text/xml; charset=utf-8

accept: text/xml, text/*

accept-charset: utf-8, iso_8859-1
```



Note. Non-XML text and latin1 will also be accepted.

Compressing the XML Message

The PeopleSoft subscribing node can accept XML data in either a compressed and encoded or an uncompressed and unencoded format. Note the following:

- The compression algorithm that PeopleSoft accepts is base64(deflate). See Uncompressing Messages for more information. When compressed, the encoding attribute of the <data> element will contain information about how many bytes each of the compressed routines produced.
- The length attribute of the <data> element is the number of Unicode characters when uncompressed. Because PeopleTools 8 and above is Unicode, two bytes represent a character that was previously represented by a single byte. One way to derive the Unicode byte length is to multiply the character length of the inflated contents of the <data> tags by 2 to get the correct length that PeopleTools can use to inflate to Unicode.
 - For example, the following code describes data that contains 4126 Unicode characters when uncompressed. When deflated, the size of the data is 532 bytes and when base64 encoded, becomes 712 bytes in size:
- <data encoding="base64(deflate)" encodedlength="712(532)" length="4126">

Authenticating the Message

When the Application Messaging Gateway receives the XML post, it routes it to the destination node and delegates the responsibility of authentication to the handler.

For example, PeopleSoft's application server validates the message being posted by checking two parameters in the XML file—*from system* and *password*. The following sample code shows these two parameters:

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

```
<request>
```

```
<from system="publishing node name" password="node group password"/>
```

When the application server receives the XML message, it validates the following:

- **Publishing node name.** This is the name of the node set in the PeopleTools Application Designer.
- **Node group password.** This is the known value for the node access password. If the password is valid, the application server processes the rest of the document. If the password is invalid, the application server halts processing and issues an error message.
- **Certificate.** For information about certificates see Digital Certificates.

Ensuring Guaranteed Delivery

To determine whether your publish was successful, you can check the standard HTTP Post return codes and the PeopleSoft Reply Document *returncode*.

If the subscribing PeopleSoft system is not available at the time the third party system attempts the publish, the message will not be delivered. The third party system must queue the messages and “retry” until the subscribing system is back online.

For the third party publisher to confirm that the message has been successfully delivered, the following conditions must be met:

- The Interpreting the HTTP Status Codes for the response must be between 200 and 299
- A valid Interpreting the XML Reply Message must be returned
- The value of the Return Code Values <return> element must be 0 (GEN_OK) or 6 (GEN_DUPLICATE)

Interpreting the HTTP Status Codes

<i>HTTP Status Code</i>	<i>Type</i>	<i>The publishing system should....</i>
100-199	Informational	Mark the contract “retry”
200-299	Success	Mark the contract “done
300-399	Redirection	Mark the contract “error”
400-499+	Client error	Mark the contract “error”
500-599++	Server error	Mark the contract “retry”
600 or higher	Not defined	Mark the contract “retry”

+ An HTTP status code in the 400-499 range generally means the client is in error—for example, the URL is incorrect, the Web server ID or password is incorrect, or the XML document is not in the expected format. These errors would probably come directly from the Web server.

++ An HTTP status code in the 500-599 range means the client request is in the correct form, but something was in error “downstream”, on the Web server or the application server. The distinction between a client error and a server error is somewhat unclear, though. For example, if the client uses the incorrect name for the target node, the servlet cannot find the node in its configuration table, and will return a status code of 502-Bad Request. A code of 502 is also be returned if the node is configured incorrectly at the gateway, and then the signon to the target application server fails.



Note. In the case of PeopleSoft, an HTTP error code indicates a communication failure, not a processing error.

Interpreting the XML Reply Message

PeopleSoft sends an XML *reply document* or message as the response for a “posted” message, so that you can determine if a post is successful. In addition, the PeopleSoft *publication contract handler* implements logic to ensure a *guaranteed delivery*. Third party publishers must implement the same logic; that is, parse the *returncode* in the reply document, then evaluate the code to guarantee delivery.

The gateway servlet makes it easy for the subscription handlers (plug-ins) to handle a ping message. The gateway servlet will filter these messages out and ask a simple yes/no question to the respective handler to verify its availability.

Sample Reply Message Format

```
<reply version=8.0>

  <operations namespace="PublishSubscribe" interface="PublishSubscribeSystem">

    <invoke opnum=1 member="Publish">

      <return type="number">returncode</return>

      <variable type="object" interface="Publication">

        <publication>

          <publishingnode>NH8A</publishingnode>

          <channel>QE_PLYR_CHNL</channel>

          <publicationid>10</publicationid>

          <publishtimestamp>1999-01 01...</publishtimestamp>

        </publication>

      </variable>

    </invoke>

  </operations>

</reply>
```

```

        </invoke>

    </operations>

</reply>

```

XML Tag	XML Field	Description
<return type>	return code	Required. Return code from the subscribing system that determines if the message was delivered.
<publicationid>	Publication ID	Not required. Publication ID for the publication. If the publication ID was specified on the original post, the value will be duplicated here. If the publication ID was not specified, the value will be the generated ID from the target node.
<publishtimestamp>	Publish Timestamp	Not required. Timestamp of when the publish reached the remote node. If the remote node is a PeopleSoft system, the timestamp indicates when the publication header was inserted into the database.
<variable>	Various fields described in the XML Field Definition Table	Not required, but recommended for tracking purposes.

Return Code Values

Code	Term	What it means to the publisher...
0	GEN_OK	The message was delivered successfully.
6	GEN_DUPLICATE+	The message was already delivered successfully.
8	GEN_ERROR	The message was not delivered successfully. Resend the message.

+ GEN_DUPLICATE means that a publication with the same publication node or channel name or publication ID already exists in the target database.



Note. If the publishing node receives an error when it tries to send the publication to the remote node, the node may not know where the error occurred. The Web server may have failed while receiving the XML post, in which case the publication was never delivered. Another possibility is that the publication may have been written to the database on the target node, and the Web server failed while waiting for the response from the application server. The publishing node cannot distinguish between the two scenarios, so it resends the post message. In the second scenario, the publication message has already been written to the database, so the insert will fail with a "duplicate row" error.

Formatting Timestamps

The PeopleSoft format for all timestamps is ISO-8601:

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



Note. ISO format allows that the +/-hhmm parameter is optional, but PeopleSoft requires it. All date and time stamps in the header and in the body of the message must include the GMT offset as +/-hhmm. This ensures that the timestamp is accurate when the subscriber interprets it (who could be in a different time zone).

Providing FieldTypes

Valid <FieldTypes> tags are required for third party publishers. The PeopleSoft system is expecting FieldType information for each record and field in the message definition. This information is static and should be available as part of the Enterprise Integration PeopleBook for each Enterprise Integration Point (EIP).

Class and Type Values

The only valid value for the *class* attribute is "R" for record. Valid values for the *type* attribute are "CHAR" for character, "DATE" for date, and "NUMBER" for number.

Example of Field Types in XML File

```
<FieldTypes>

  <MASTER_ITEM_TBL class="R">

    <SETID type="CHAR"/>

    <INV_ITEM_ID type="CHAR"/>

    <ITM_STATUS_EFFDT type="DATE"/>

    <DESCR type="CHAR"/>
```



```

        <UNIT_MEASURE_STD type="CHAR"/>

        <INV_ITEM_GROUP type="CHAR"/>

    </MASTER_ITEM_TBL>

    <BU_ITEMS_WTVOL class="R">

        <BUSINESS_UNIT type="CHAR"/>

        <INV_ITEM_ID type="CHAR"/>

        <UNIT_OF_MEASURE type="CHAR"/>

        <PACKING_WEIGHT type="NUMBER"/>

        <PACKING_VOLUME type="NUMBER"/>

    </BU_ITEMS_WTVOL>

</FieldTypes>

<MsgData>

<Transaction>

data

</Transaction>

</MsgData>

```

Subscribing to PeopleSoft

Subscribing to PeopleSoft data requires that you be familiar with XML format and some parts of the PeopleSoft Application Messaging system. The following table lists a few basic requirements:

<i>What you need...</i>	<i>Where to find it...</i>
Ability to receive an XML string on an HTTP port, and to parse and process this data.	Various Web sites that explain XML and the Administering the Application Messaging Gateway PeopleBook.
Knowledge of XML fields in the header section of the XML message.	Interpreting FieldTypes
Understanding of data elements of the PeopleSoft Application Message Definition.	Sample messages provided with the PeopleSoft Integration SDK installed with your application. Application-specific XML Fields and the “EIP Catalog” section of the <i>Enterprise Integration PeopleBook</i> .
An application messaging handler.	Simple File Publication Handler included

	with your PeopleSoft application, plus Java source code included in the PeopleSoft Integration SDK installed.
Support of a PeopleSoft test environment.	PeopleSoft Alliances, http://www.peoplesoft.com/

Steps to Subscribe from PeopleSoft

There are three basic steps to subscribe to PeopleSoft data:

- Creating message components in PeopleSoft
- Modifying the PeopleSoft gateway handler
- Creating your message

Creating Subscription Message Components

Using PeopleSoft's Application Designer, do the following or verify that it has been done:

- Identify the local node of the system from which you are subscribing (source node)
- Define the remote node that is subscribing to the message (destination node)
- Determine the message channel routing rules
- Verify there are message definitions
- Develop the custom subscription handler

Configuring the Handler

You must add and configure the destination node, that is, the subscribing node to the PeopleSoft handler.



For more information about configuring nodes to the gateway, see Messaging Server Administration.

Before You Subscribe

This section describes information you need as you subscribe to PeopleSoft messages.

- Uncompressing Messages
- Interpreting FieldTypes

- XML File Format
- Configuring an External Handler

Uncompressing Messages

All messages from PeopleSoft are compressed in base64(deflate). Prior to posting, the XML data is converted to UTF-8 and compressed using deflate and base64. This is to make the binary data suitable for embedding in an XML document, and safe for transmitting through 7-bit ASCII programs.

To uncompress messages, you currently need to decode (base64) the data first, then inflate the results of the base64 decode. The attributes of the <data> element describe the format of the data:

```
<publicationdataversion>

  <version>VERSION_01</version>

  <data

    length="identity-length"

    encoding="base64(deflate)"

    encodedlength="base64-encoded-length(deflated-length)"

    >deflated, base-64 encoded XML data</data>

  </publicationdataversion>
```

Interpreting FieldTypes

Publications from a PeopleSoft system include FieldType information for each record and field in the message definition. As a subscriber, you can use this information to validate the data type for each field. This information is static and included in the *Enterprise Integration PeopleBook* for each Enterprise Integration Point (EIP).

Class and Type Values

The only valid value for the *class* attribute is “R” for record. Valid values for the *type* attribute are “CHAR” for character, “DATE” for date, and “NUMBER” for number.

Example of Field Types in XML File

```
<FieldTypes>

  <MASTER_ITEM_TBL class="R">

    <SETID type="CHAR"/>
```

```

        <INV_ITEM_ID type="CHAR"/>

        <ITM_STATUS_EFFDT type="DATE"/>

        <DESCR type="CHAR"/>

        <UNIT_MEASURE_STD type="CHAR"/>

        <INV_ITEM_GROUP type="CHAR"/>

    </MASTER_ITEM_TBL>

    <BU_ITEMS_WTVOL class="R">

        <BUSINESS_UNIT type="CHAR"/>

        <INV_ITEM_ID type="CHAR"/>

        <UNIT_OF_MEASURE type="CHAR"/>

        <PACKING_WEIGHT type="NUMBER"/>

        <PACKING_VOLUME type="NUMBER"/>

    </BU_ITEMS_WTVOL>

</FieldTypes>

<MsgData>

<Transaction>

data

</Transaction>

</MsgData>

```

XML File Format

The XML document for a publication uses a specific format, shown here. The values of the fields in *Italics* can vary for each message. This is the format for sending a compressed, encoded message.

```

<?xml version="1.0"?>

<request version="peopletools-version">

    <to node="destination-node-name"/>

    <from node="source-node-name" password="source-node-group-password"/>

    <operations namespace="PublishSubscribe"interface="PublishSubscribeSystem">

        <invoke member="Publish">

```

```

<variable type="object" interface="Publication">
    <publication>
        <publishingnode>source-node-name</publishingnode>
        <channel>channel-name</channel>
        <publicationid>publication-id</publicationid>
        <subchannel>subchannel</subchannel>
        <subject>message-name</subject>
        <subjectdetail>message-detail</subjectdetail>
        <originatingnode>originating-node-name</originatingnode>
        <publisher>publishing-operator-id</publisher>
        <publicationprocess>publication-process</publicationprocess>
        <publishtimestamp>publish-timestamp</publishtimestamp>
        <status>publication-status</status>
        <defaultdataversion>default-message-version-name</defaultdataversion>
        <dataversions>
            <publicationdataversion>
                <version>VERSION_1</version>,
                <data length="identity-length"
                    encoding="base64 (deflate) "
                    encodedlength="base64-encoded-length(deflated-length)"
                    >deflated, base-64 encoded XML data</data>
                </publicationdataversion>
            </dataversions>
        </publication>
    </variable>
</invoke>
</operations>
</request>

```



Note. A message can contain multiple transactions and the <MsgData> label contains transactions for the message. For example, if the message is *PURCHASE_ORDER_SYNC* it can contain multiple Purchase Order Headers and their associated Purchase Order Lines. Therefore, the <Transaction> label could occur multiple times, and contain the specific elements of the application message definition and their values.

XML Field Definition Table

This table describes the XML fields in the file.

<i>XML Tag</i>	<i>XML Field</i>	<i>Description</i>
<request version=	"Peopletools version"	Required. PeopleTools version for which this XML file is valid, for example, "8.00".
<to node=	"destination node name"	Required. Name of the node for which the message is intended. For third party publishers, this must correspond to an entry in the node lookup table on the gateway servlet, and the name of the local node (node definition) on the receiving PeopleSoft system. For PeopleSoft publishers, this is the name of the node definition for the third party system as defined in the sending PeopleSoft system.
<from node= <publishingnode>	"source node name"	Required sometimes. Name of the node which published the message. The fields {publishing node, channel, publication ID} uniquely identify the publication. For third party publishers, this node name must match the node definition for the third party system as defined in the receiving PeopleSoft system. This field is in two places: <from.node> and <publishingnode>: <from.node> is required and <publishingnode> is not required. However, if <publishingnode> is present, it must match the value in <from.node>.

password=	“source node group password”	<p>Required sometimes. Clear text password associated with the destination node. This value is stored in the PeopleSoft database and must be communicated to the system administrators for the publishing system.</p> <p>If the node definition on the sending system has a node group defined, the password will be present. If the node definition on the receiving system has a node group defined, the password must be present and must match the node group password.</p>
<channel>	Channel name	<p>Required. Name of message channel containing the message. The fields, publishing node, channel, and publication ID uniquely identify the publication.</p>
<publicationid>	Publication ID	<p>Not required. System-generated identifier for the publication. The fields, publishing node, channel, and publication ID uniquely identify the publication.</p> <p>If the <publishing node name> is specified and the <publication ID> is omitted, the publication ID is set to the next available publication ID on that channel within the subscribing PeopleSoft database.</p>
<subchannel>	Subchannel	<p>Not required. Name of subchannel which contains the message. Messages in the same channel but in different subchannels are assumed to refer to distinct objects, for example, different POs or different employees. They will be processed in parallel where possible.</p> <p>This field should contain the concatenated values that represent the subchannel. For example, if the subchannel is Business Unit\ Journal ID, then the value of this field is M04123456789 (where Business Unit = M04 and Journal ID = 123456789).</p> <p>For third party publishers, this field</p>

		<p>should be included if the subscribing PeopleSoft system has a subchannel defined. Otherwise, it may be omitted.</p> <p>For PeopleSoft publishers, this field will be present if the subchannel is non-null. If there is no subchannel for the publication, there is no entry in the XML file.</p>
<subject>	Message name	Required. Name of the message as defined in the PeopleSoft system.
<subjectdetail>	Message detail	Not required. Application-defined subtype of message name. Application may require it.
<originatingnode>	Originating node name	Not required. Name of the node that originally published the message and used to prevent circular publishes. If not in XML file, the system sets it to the publishing node name.
<publisher>	Publishing operator ID	Not required. Application-defined operator ID/class that published the message. Application may require it.
<publicationprocess>	Publication process	Not required. Application-defined name of the program that generated the message. Application may require it.
<publishtimestamp>	Publish timestamp	<p>Time when publish occurred (in PeopleSoft systems, this means when the publication was actually written to the database). If it's not present, the receiving system will set it when the publication is written to the database.</p> <p>Format must be ISO-8601 date/time notation:</p> <p>CCYY-MM-DDTHH:MM:SS.sssss[+/-hhmm]</p>
<status>	Status	Not required. Status of the publication. Informational only; ignored by the pub/sub system in all cases.
<defaultdataversion>	Default message version name	Not required. Default message version for the sending system. Third party publishers can omit this field—the default version may be

		reset by the receiving system.
<messageversion>	Message version name	Required. Name of the message version. For publications with multiple data versions, there will be multiple occurrences of the data.
<data.length>	"identity-length"	Required. Specifies the number of Unicode characters of the data before any encodings have been applied. Also known as the identity encoding.
<data.encoding>	"base64(deflate)"	<p>Not required. Specifies the set of encodings and their order applied to the publication data. A value of "X(Y)" indicates that encoding Y was applied to the raw data, and encoding X was applied to the result of Y. For PeopleTools 8, the only valid value for the encoding attribute is "base64(deflate)" (no spaces, all lower case). For posted messages coming into PeopleSoft, if the encoding attribute is not present, the data is assumed to be uncompressed.</p> <p>Not required, but will always be present for messages from PeopleSoft.</p>
<data.encodedlength>	"base64-encoded-length(deflated-length)"	<p>Required sometimes. Specifies the length of the data after each encoding specified in encoding has been applied. Required if data.encoding is present. Will always be present for messages from PeopleSoft.</p> <p>Refers to the number of characters, not the number of bytes, to which the text is encoded.</p>
<data>	Data blob or message data	Required. Message-specific XML data elements and values. For a compressed message, the compressed "blob" follows this tag. For an uncompressed message, the <MsgData> and <Transaction> tags follow.
<MsgData>	Message data	Required. Not seen in the above example, but seen in the XML Transaction sample section. This tag is required to show where the content

		of the message starts. This tag can include multiple <Transaction>s.
<Transaction>	Separator for message data transactions	Required. Not seen in the above example, but seen in the XML Transaction sample section. This tag is required to show where each transaction in the message starts. There will be one or many <Transaction> tags.

Application-specific XML Fields

The following table shows an application-specific subset of the XML fields in the XML Field Definition Table. The third party publisher will need to supply these. Some of the fields are static, that is, the same for all publications of a message; while others depend on the specific implementation.

XML field	Static?	Where to get the value...
Destination node name	No	When the integration point is implemented at a customer site (or in a test environment), set this field to the node name of the local node of the PeopleSoft system to which you are publishing.
Source node name	No	When the integration point is implemented at a customer site (or in a test environment), set this field to the node name of the remote node that is defined in the PeopleSoft application which represents your application.
Channel name	Yes	EIP documentation supplies this channel name; however, users can modify it.
Subchannel	Yes	EIP documentation supplies this subchannel name; however, users can modify it.
Message name	Yes	EIP documentation supplies this message name; however, users can modify it.
Message detail	Yes	EIP documentation supplies this message name detail; however, users can modify it.
Originating node name	No	Same as Publishing node name for third party publishers (see Publishing node name).
Publishing operator ID	No	EIP documentation supplies the application-specific requirements for this publisher name, if there are any.
Message version name	Yes	Name of the message version. Standard for this PeopleTools release is VERSION_1, as shown in sample.
Data	No	Message content, based on the PeopleSoft Application Message Definition. Each message

		definition is detailed in the <i>Enterprise Integration PeopleBook</i> , and will be referred to as an Enterprise Integration Point (EIP). To understand the message data tags and how to set them, read all of the appropriate Enterprise Inntegration Point documentation.
--	--	--

Common Application Messaging Attributes (PSCAMA)

PSCAMA, or *PeopleSoft Common Application Messaging Attributes*, is an XML tag containing fields common to all messages. The PSCAMA tag repeats for each row in each level of the <Transaction> section.

PSCAMA is used primarily so the publisher can give the subscriber important information about the message data. The PSCAMA tells the subscriber in what language the data is, and also the action taken on the row of data sent; for example, was the action an Add, Update, or Delete in the publishing system.



Note. As a third party **publisher**, be aware that the PeopleSoft subscription process is expecting the PSCAMA values to be set. Conversely, as a third party **subscriber**, be aware of what the PSCAMA values mean, and what you may expect to see in different types of messages (for example full data vs. incremental data).

PSCAMA Record Characteristics

- **Size.** The size of the PSCAMA record varies—in particular, notice a difference between the first PSCAMA record and the ones that follow. The first PSCAMA contains all of the fields, while, the other PSCAMA records only have the AUDIT_ACTN field. This is because the AUDIT_ACTN field is the only field that *can* change for each row of data. Every message will contain the first PSCAMA with all of the fields.
- **Absence of PSCAMA Record.** Although the first PSCAMA record is always present in a message, not all of the remaining PSCAMA records are sent in the message. If a <PSCAMA> tag is not included for a specific row in a message, then you can assume that the row has not changed. Therefore, the PSCAMA tag should be present on any row that has changed. In addition, if the PSCAMA <AUDIT_ACTN> tag is blank (<AUDIT_ACTN>” ”</AUDIT_ACTN>) or null (<AUDIT_ACTN></ AUDIT_ACTN> or <AUDIT_ACTN />);you can also assume that the row has not changed.
- **Sample Scenarios.** Note the results of the following situations when PeopleSoft publishes a message and the effect on PSCAMA.

Scenario	Message includes...	Status of Row-level PSCAMA
Full Data Publish	All rows in the transaction	Each row has a PSCAMA record with an AUDIT_ACTN = ‘A’

Scenario	Message includes...	Status of Row-level PSCAMA
Incremental Publish with CopyRowset	All rows in the transaction	Only rows that have changed have PSCAMA record
Incremental Publish with CopyRowsetDelta	Only rows that have changed and their parents and grandparents	Only rows that have changed have PSCAMA record



Note. If possible, third party publishers should support the same type of publish results.

PSCAMA Record Fields

The following screen shows the PSCAMA record, and the PSCAMA Field Table gives the definition for each field.

Num	Field Name	Type	Len	Format	H	Short Name	Long Name
1	LANGUAGE_CD	Char	3	Upper		Lang Cd	Language Code
2	AUDIT_ACTN	Char	1	Upper		Action	Action
3	BASE_LANGUAGE_CD	Char	3	Upper		Lang Cd	Language Code
4	MSG_SEQ_FLG	Char	1	Upper		Message Seq	Message Sequence
5	PROCESS_INSTANCE	Nbr	10			Instance	Process Instance
6	PUBLISH_RULE_ID	Char	30	Upper		Publish Rule	Publish Rule ID
7	MSGNODENAME	Char	15	Upper		Msg Node	Message Node Name

PSCAMA Record

PSCAMA Field Table

Field Name	Description
LANGUAGE_CD	Required. Language code in which the message is published. When publishing from components, the system sets this field to the operator's default language code. The application developer can override this if necessary.
AUDIT_ACTN	Required. Audit Action code which identifies the row of data as an Add, Update, or Delete.
BASE_LANGUAGE_CD	Not required. Base language code of the publishing database. Used by the generic full table subscription PeopleCode to help determine which tables to update.
Message Sequence Flag	Not required. Indicates whether the message is a header (H), trailer (T) or contains data (blank). The subscribing database can use this field to initiate processes. For example, the header

	message may cause staging tables to be cleared while the trailer would indicate that all the data has been received and a update job should be initiated
PROCESS_INSTANCE+	Not required. Process Instance of the batch job which created the message. This can be used by the subscribing database along with the publishing node and publication ID to uniquely identify a group of messages from the publishing node.
PUBLISH_RULE_ID+	Not required. Use as an audit field to indicate which Publish Rule was used to create the message, and by routing PeopleCode to locate the chunking rule used. Then the chunking rule determines to which nodes the message gets published.
MSGNODENAME	Not required. The node to which the message will be published. This is an optional field which is passed to the Publish utility by the application AE program. Routing PeopleCode must be look for a value in this field and return it to the application server.

+Third party applications should ignore these fields.

Language Code

Each message can contain only one Language Code and it appears on the first PSCAMA record. As a publisher, you will need to set this code to match the Language Codes as they are defined in the PeopleSoft database and listed in the following table. As a subscriber, you will need to interpret these codes to determine the language of the message data.

<i>Language</i>	<i>Code</i>
Brazilian Portuguese	POR
Canadian French	CFR
Dutch	DUT
English	ENG
French	FRA
German	GER
Italian	ITA
Japanese	JPN
Spanish	ESP



Note: There can be only one language code for the entire message. If you need to publish messages in multiple languages, then you must publish multiple messages. If you are receiving a message that has incremental changes, only the rows that have changed *and their parents* are present in the message.

Audit Action Codes

Rows in the message have an audit action field to denote the type of transaction. The Audit Action is required so that the subscribing system knows how to process the incoming data.

The audit action values match those used in PeopleSoft's audit trail processing, as shown in the following table:

Code	Description
A	Add row (insert)
D	Delete row
C	Change row (updated, but no key fields changed). The system writes old values to the audit table.
K	Change row (updated and at least one key field changed). The system writes old values to the audit table.
N	Change row (updated and at least one key field changed). The system writes new values to the audit table.
“ ”	Blank value means that nothing in that row has changed. This is the default, and the value used to tag the parent rows of children that have changed.

Message Sequence Flag

Large transactions may span multiple messages. In these cases, the PeopleSoft system uses Header and Trailer messages to indicate the start and end of the set of messages. The Header and Trailer messages do not contain any message data—the messages in between the Header and the Trailer contain the message data.

Third party publishers may be required to send Header and Trailer messages, depending on the specific EIP (Enterprise Integration Point).

Valid <MSG_SEQ_FLG> values are *H* for Header and *T* for Trailer.

Developing a Subscription Handler

The gateway ensures that messages get routed to the correct subscription handler. You can use the Simple File Publication Handler as a guide when creating your own subscription handler. It has the following features:

- Configuration screens that allow the user to add, edit, or delete nodes and enter other parameters
- Maintains a list of registered nodes

Simple File Publication Handler

This section describes the classes of the Simple File Publication Handler.

FileHandlerEntry.class

The FileHandlerEntry Java class is responsible for reading and writing the actual configuration parameters to and from the defined configuration file. The File Handler Entry Java program writes the configuration file as a comma-delimited file.

XSLHandlerEntry.class

The XSLHandlerEntry Java class is a clone of the FileHandlerEntry Java class. Since the header, decode, and uncompress options were removed, it was necessary to eliminate the need for the File Handler Entry to read and write these respective values to a configuration file. This is because PeopleSoft provides utility classes to uncompress the data inside the publication message.

SimpleFileHandler.class

The SimpleFileHandler is the Java class that is responsible for loading and unloading your actual handler into memory. By loading the handler, the handler becomes active to receive publications from the PeopleSoft Gateway. In turn, pressing "Unload" will remove the particular handler from the list of active registered handlers. This allows you to disengage your handler from the pub/sub system temporarily without unloading the entire gateway servlet.

Simple XSL Handler

The Simple XSL Handler Java class was designed to have the overall functionality and same user interface for loading, unloading and configuration as the Simple File Handler. The Simple File Handler class was cloned to create the Simple XSL Handler. Each file name was also renamed in the Simple XSL Handler to make the appropriate calls to XSLEntry, AdministerXSLHandler, AdministerXSLHandlerAddMode, AdministerXSLHandlerEditMode, and AdministerXSLHandlerDeleteMode.

The following is an example of invoking a new instance of the Simple XSL Publication Handler:

```
Entry entry = (Entry)elements.nextElement();

m_pubHandlers.put( entry.getNodeName(),

    new SimpleFilePublicationHandler( entry, 0 ) );
```

With

```

XSLEntry entry = (XSLEntry)elements.nextElement();

    m_pubHandlers.put( entry.getNodeName(),

        new SimpleXSLPublicationHandler( entry, 0 ) );

```

It will also be necessary to change the actual name of the configuration file where all values are being stored. The names of the configuration files are defined in the Simple File Handler Constants Template.

The following is an example of registering the appropriate name for the configuration file:

```

BufferedReader config = new BufferedReader( new FileReader( "filehandler.cfg" )
);

    if ( config.ready() )

        m_configFile = config.readLine();

```

With

```

BufferedReader config = new BufferedReader( new FileReader( "xslhandler.cfg" )
);

    if ( config.ready() )

        m_configFile = config.readLine();

```

Administer File Handler

The Administer File Handler is the Java class responsible for rendering the entire Web content for the Simple File Handler Directory. The Simple File Handler Directory is displayed in a form of a table defined by simple table nodes and widths. The table includes the "Edit", "Delete" and "Add" button, in addition to the pre-selected values from the user. It is important to note that when the user presses one of the buttons, the Simple File Handler is responsible for creating a new instance of Administer File Handler Add Mode, Administer File Handler Delete Mode, or Administer File Handler Edit Mode. Each of these Java classes renders their own content similar to the Administer File Handler Java program.

Simple File Handler Directory

Node Name	Output Directory	Include Header?	Uncompress?	Base64 Decode?	Edit	Delete
QE_LOCAL	c:\temp\file	No	No	No	Edit	Delete

Add a file handler node

[Back to Handler Directory](#)

Simple File Handler Directory

Administer XSL Handler

The Administer XSL Handler is a clone of the Administer File Handler class. The Administer File Handler offers a display table that is identical to the one used in the Simple XSL Handler. However, remember that the need for the user to select the header, uncompress, and decoded options was removed. The table nodes and widths were essentially reduced, by 3, to eliminate these text fields. All instances of these variables that even referenced to the header, compress or decode options were removed altogether.

The following is an example of changing the number of nodes and widths to accommodate the Simple XSL Handler Directory:

```

if ( readOnly )
{
    nodes = new Node[2]; //5 Original Value
    widths = new String[2]; //5 Original Value
}
else
{
    nodes = new Node[4]; //7 Original Value
    widths = new String[4]; //7 Original Value
}

```

The following is an example of changing the assignment of objects into the nodes and widths:

```

nodes[2] = doc.createTextNode( "Edit" ); // Original Value of 5
widths[2] = "5%"; // Original Value of 5
nodes[3] = doc.createTextNode( "Delete" ); // Original Value of 6
widths[3] = "5%"; // Original Value of 6

```

Administer File Handler Add Mode

The Administer File Add Mode Java class is very similar to the Administer File Handler Java program. It creates a user friendly display table that allows the user to view configuration options and sets default values for appropriate user input. The Administer File Add Mode Java program is invoked when the user selects to "Add a file handler node" from the Simple File Handler Directory. It is actually called from the Simple File Handler.

Administer XSL Handler Add Mode

The Administer XSL Handler Add Mode is a clone of the Administer File Handler Add Mode. Again, it was a simple process eliminating the need for the user to select the header, uncompress, and decoded options. To eliminate any need for these input options, the table's node and width definitions was reduced by three. All instances of variables that even referenced to the header, compress or decode options were removed altogether.

Administer File Handler Edit Mode

The Administer File Edit Mode Java class is very similar to the Administer File Handler Java program. It creates a user friendly display table that allows the user to view configuration options. The Administer File Edit Mode Java program is invoked when the user selects to "Edit" a node from the Simple File Handler Directory. It is actually called from the Simple File Handler Java class.

Administer XSL Handler Edit Mode

The Administer File Handler Edit Mode is a clone of the Administer XSL Handler Add Mode. The table nodes and widths were essentially reduced, by 3, to eliminate the text fields and input options. All instances of variables that even referenced the header, compress or decode options were removed.

Simple File Publication Handler Class

The Simple File Publication Handler is the Java class responsible for the actual processing of the publication message. It receives the publication message through a Subscription Interface called the invokeHandler. This function is called when a publication is received by the gateway servlet and is determined, by the gateway servlet, that this particular handler is the proper recipient of the publication. The publication message is handed to the handler as an input parameter to the invokeHandler function by way of an instance of the PublicationMessage class. With this, the Simple File Publication Handler has the ability to create a response to inform that it is willing to accept data.

The following function is called when a publication is received by the gateway servlet. The servlet determines that this particular handler is the proper recipient of the publication.

```
public String invokeHandler(PublicationMessage pubMsg, String clientDN)
    throws PublicationHandlerException
{
}
```

Before the flat file is written, the user does have several optional parameters to choose from, which are selected from the Simple File Handler Interface. For example, the Simple File Publication Handler will write the PeopleSoft application message to any output directory specified by the user on the PeopleSoft Gateway interface.

All optional parameters are currently retrieved through the Entry Java class. With this, the Simple File Publication Handler uses IF/ELSE logic to determine which optional parameters are selected by the user on the Simple File Handler Interface.

```

if ( m_objEntry.retrieveHeader() )

    {

        publication = pubMsg.getPublicationMessage();

        out.write( publication.getBytes() );

    }

else if ( m_objEntry.base64DecodeData() )

    {

        publication = pubMsg.getPublicationData( false );

        String encLengths = pubMsg.getAttributeValue( "data",
"encodedlength" );

        int openParen = encLengths.indexOf( '(' );

        int base64Length = Integer.parseInt( encLengths.substring( 0,
openParen ) );

        int closeParen = encLengths.indexOf( ')' );

        int deflateLength =
Integer.parseInt(encLengths.substring(openParen+1, closeParen));

        byte[] decodedPub = doBase64Decode( publication, base64Length, deflateLength );

if ( m_objEntry.inflateData() )

    {

        String rawLength = pubMsg.getAttributeValue( "data", "length" );

        int uncompressedLength = Integer.parseInt( rawLength );

        String cleartextPub = doInflate( decodedPub, deflateLength,
uncompressedLength );

        out.write( cleartextPub.getBytes() );

    }

```

```

else
{
    out.write( decodedPub );
}
}
else
{
    publication = pubMsg.getPublicationData( false );
    out.write( publication.getBytes() );
}

```

Once the message has been processed it's written to a flat file located in the appropriate directory. Each message is written to a new unique flat file named after the unique publication ID, and message name. The Simple File Publication Handler writes the message to a flat file.

```

String pubId = pubMsg.getTagValue( "publicationid" );
String msgName = pubMsg.getTagValue( "subject" );
String outputFile = m_objEntry.getOutputDirectory() +
    System.getProperty( "file.separator" ) +
        msgName + "_" + pubId + ".xml";

out = new FileOutputStream( outputFile );
out.write( cleartextPub.getBytes() );

```

Finally, a response is sent to the publishing node indicating that the message was processed. The response message comes from the Publication Message Reply Java class, passing in the appropriate unique values needed. The Simple File Publication Handler creates a new instance of the Publication Message Reply Java class, setting values as needed.

```

PublicationMessageReply pubMsgReply = new PublicationMessageReply
("SimpleFilePublicationHandler", 0);

pubMsgReply.setPublishingNode(pubMsg.getTagValue( "publishingnode" ));
pubMsgReply.setChannel(pubMsg.getTagValue( "channel" ));
pubMsgReply.setPublicationId(pubMsg.getTagValue( "publicationid" ));

```

```
pubMsgReply.setPublishTimeStamp(pubMsg.getTagValue( "publishtimestamp"));
```

The Simple File Publication Handler will send a reply document to the publishing node stating that the publication message has been processed

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

<reply>

  <operations namespace="External Gateway Servlet"
  interface="SimpleFilePublicationHandler">

    <invoke opnum="1" member="invokeHandler">

      <return type="number">0</return>

      <variable type="object" interface="publication">

        <publication>

          <publishingnode>QE_LOCAL</publishingnode>

          <channel>BUS_EXP_MSG_CHNL</channel>

          <publicationid>71</publicationid>

          <publishtimestamp>2000-06-05T17:11:15.413000-0700</publishtimestamp>

        </publication>

      </variable>

    </invoke>

  </operations>

</reply>
```

Simple XSL Publication Handler

The Simple XSL Publication Handler automatically stripes headers, inflates, and decodes the PeopleSoft Application Message before processing

```
int openParen = encLengths.indexOf( '(' );

int base64Length = Integer.parseInt( encLengths.substring( 0, openParen ) );

int closeParen = encLengths.indexOf( ')' );

int deflateLength = Integer.parseInt( encLengths.substring( openParen+1,
closeParen ) );
```

```

byte[] decodedPub = doBase64Decode( publication, base64Length, deflateLength );

try {

    String rawLength = pubMsg.getAttributeValue( "data", "length" );

    int uncompressedLength = Integer.parseInt( rawLength );

    String cleartextPub = doInflate( decodedPub, deflateLength, uncompressedLength
    );

    String strippedText = cleartextPub.substring(21);

    String strippedTextRight = strippedText.trim();

    XSLTProcessor processor = XSLTProcessorFactory.getProcessor();

    processor.process( new XSLTInputSource( new
    java.io.StringReader( strippedTextRight ) ),

        new XSLTInputSource( Directory + ".xsl" ),

        new XSLTResultTarget( Directory + "_" +
    pubId + ".html" ) );

}

```

The Simple XSL Publication Handler generates a response as a unique reply document to the publishing node indicating that the message has been processed.

```

PublicationMessageReply pubMsgReply = new PublicationMessageReply

("SimpleXSLPublicationMessage", 0);

pubMsgReply.setPublishingNode( pubMsg.getTagValue( "publishingnode" ) );

pubMsgReply.setChannel( pubMsg.getTagValue( "channel" ) );

pubMsgReply.setPublicationId( pubMsg.getTagValue( "publicationid" ) );

pubMsgReply.setPublishTimeStamp( pubMsg.getTagValue( "publishtimestamp" ) );

```

SimpleFileHandler Java Source Code

The PeopleSoft Integration SDK installed with your application includes Java source code and class files for the Simple File Publication Handler, which you can modify for your own purposes.

The Java source code can be found in:

```
<PS_HOME>\sdk\psappmsg\src\java\samples\simple_file_handler
```

The class files can be found in:

```
<PS_HOME>\sdk\psappmsg\classes
```



For more information about the application messaging resources in the PeopleSoft Integration SDK, see Application Messaging SDK.

Samples

This section includes the following samples:

- XML Transaction
- Posting XML to an Application Messaging Gateway
- Third Party to PeopleSoft XML/HTTP Conversation
- PeopleSoft to Third Party XML/HTTP Conversation
- Posting from a C Program

XML Transaction

The PeopleSoft Integration SDK includes a sample application message, SDK_BUS_EXP_APPR_MSG. Depending on how you intend to use the transaction, you may need to include a PeopleSoft application message header, and you may want to compress the message content. The following listing includes examples of each feature, in the order they would appear in the message, extracted from the samples installed with the integration SDK.



Samples displayed here are formatted with newlines and indentation for easy viewing, but XML doesn't require such formatting.



For more information about the sample messages installed with the integration SDK, see SDK Resources.

XML processing instruction tag

This is required for all messages:

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

PeopleSoft application message header

Two of the sample messages delivered with the integration SDK have a message header:

- sdk_nopsxmlpost_comp_msg.xml
- sdk_nopsxmlpost_uncomp_msg.xml

Following is the sample message header:

```
<request version="8.10-N5">
  <to node="PS_DESKTOP"/>
  <from node="THIRD_PARTY" password="" />
  <operations namespace="PublishSubscribe" interface="PublishSubscribeSystem">
    <invoke member="Publish">
      <variable type="object" interface="Publication">
        <publication>
          <publishingnode>THIRD_PARTY</publishingnode>
          <channel>SDK_BUS_EXP_MSG_CHNL</channel>
          <publicationid>37</publicationid>
          <subchannel></subchannel>
          <subject>SDK_BUS_EXP_APPR_MSG</subject>
          <subjectdetail></subjectdetail>
          <class>PSAPMSG</class>
          <originatingnode>THIRD_PARTY</originatingnode>
          <publisher>3RD_PARTY_CNTCT</publisher>
          <publicationprocess>3RD_PARTY_SYS</publicationprocess>
          <publishtimestamp>2000-07-21T14:36:11.977000-0700</publishtimestamp>
```



```

<status>4</status>

<defaultdataversion>VERSION_1</defaultdataversion>

<dataversions>

  <publicationdataversion>

    <version>VERSION_1</version>

  <data>

```

Uncompressed application message content

The message content can be compressed or uncompressed.

You don't have to provide the message header if you publish it using the PeopleSoft XML Poster utility, which generates it based on the parameters you provide. Two of the sample messages delivered with the integration SDK have no header—they contain just the transaction content. They're identical, except the second message is formatted for viewing in a text editor or word processor:

- sdk_bus_exp.xml
- sdk_bus_exp_formatted.xml

Following is the uncompressed sample transaction content:

```

<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"/>

    <PUBLISH_RULE_ID type="CHAR"/>

    <MSGNODENAME type="CHAR"/>

</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>

        <PUBLISH_RULE_ID></PUBLISH_RULE_ID>

        <MSGNODENAME></MSGNODENAME>

    </PSCAMA>

</Transaction>

</MsgData>

</SDK_BUS_EXP_APPR_MSG>

```

Compressed application message content

Your message needs a header if you compress the message content, but PeopleSoft doesn't require compressed data. One of the sample messages delivered with the integration SDK contains content compressed with Base64 compression:

- sdk_nopsxmlpost_comp_msg.xml

For compressed data, the enclosing <data> tag should include several compression attributes. Following is a sample of the compressed transaction content enclosed in the <data> tags:

```

<data encoding="base64 (deflate)" encodedlength="444 (332)"
length="954">eJx1k09PwzAMxb/K1MtuFO7dJjfxuookDXGC4BRNMKFJ44/ohODbk20dbJ17i/rzc99
znGL2/boZfa0+2/X722R8c3U9nk0LkrexDBTxwUaw1kVN1bSYr1ebZ//zsWrPKyy60dNm2baTzGUHhNq
qWo62qXiSiQW4LO/AoT5K30EJHo+QQqlrH+eqYpS18WoA7S2SBx9oiNYmcSOw4yboEv8qJJJwjFI0WqP
x/aZ5L/yxiavvUXXKTiE5I9sfHJhQ0ywWk0TcS1d4iZ6P763k76VXfxk5UIT/2JEFDGMXAhe0haM9bDM6
hEY+8eufIBmcbQoZKtP5iS/JekmlhSYCG00QKTBWgYixDkGmDQHjTAyWkEMO6tOCR8I5ZMOsagUTDG2R
DqWpaRBcUxos4qbFJN2hA9yeQH3Klw+nD0u2LXG6X6ev/iX2OvwTlGeo=</data>

```



For more information about compressing message data, see [Compressing the XML Message](#).

Header closing tags

Messages with a header require these closing tags:

```

        </data>

    </publicationdataversion>

</dataversions>

</publication>

</variable>

```

```

        </invoke>

    </operations>

</request>

```

Posting XML to an Application Messaging Gateway

A file called *xmlexport.html* is available to help you post. Copy this file to your local directory, and launch it in your Web browser.

To test your XML file

1. Enter the path and file name of the XML file you want to post.

You may use the **Browse** button to search for the file.

2. Enter the URL of the Application Messaging Gateway, for example, <http://....psft.pt8.gateway.GatewayServlet>
3. Select **post** to execute the HTTP Post of your XML file. This will publish the message to the subscribing PeopleSoft system.

If the post is unsuccessful, check for errors.

Possible Posting Errors

Note the following errors that could occur during your testing.

<i>Xmlexport.html will return...</i>	<i>Possible Cause</i>
502 Bad Gateway	App Server for the Destination (Subscribing) Node is down
	The <publicationid> specified is a duplicate ID
An error indicating the page can not be displayed or is unavailable	Application Messaging Gateway is down
412 Precondition Failed	Node name is not defined on the Gateway (i.e. <to system="JUNK"/>)
An error	The <publishingnode> not known to subscribing node (i.e. <publishingnode>JUNK</publishingnode>)
An error	The <publishingnode> == Local Node of the subscribing system

Third Party to PeopleSoft XML/HTTP Conversation

This section shows the specific steps, with XML file samples, needed for a third-party application to publish data to a PeopleSoft application. (Step 1 and 2 are optional).

1. The third party application pings the remote PeopleSoft node with an HTTP Request.



Note. The response header variable *content type* must be set to *text/xml*.

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

<request version="1.0">

  <from node="THIRD-PARTY" />

  <to node="PEOPLESFT" />

  <operations namespace="PublishSubscribe" interface="PublishSubscribeSystem">

    <invoke member="PingNode">

      <variable type="string">PEOPLESFT</variable>

    </invoke>

  </operations>

</request>
```

2. PeopleSoft node replies to the ping, as in this reply document:

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

<reply>

  <operations namespace="PublishSubscribe" interface="PublishSubscribeSystem">

    <invoke opnum="1" member="PingNode">

      <return type="number">0</return>

    </invoke>

  </operations>

</reply>
```

3. The third-party publishes a message to the PeopleSoft database with a new HTTP request (Post).

```

<?xml version="1.0" ?>

  <request version="1.0">

    <from node="THIRD-PARTY" />

    <to node="PEOPLESOFT" />

    <operations namespace="PublishSubscribe"
interface="PublishSubscribeSystem">

      <invoke member="Publish">

        <variable type="object" interface="Publication">

          <publication>

            <publishingnode>THIRD-PARTY</publishingnode>

            <channel>ENTERPRISE_SETUP</channel>

            <publicationid>38</publicationid>

            <subject>CURRENCY_SYNC</subject>

            <class>PSAPMSG</class>

            <originatingnode>THIRD-PARTY</originatingnode>

            <publisher>VP1</publisher>

            <publicationprocess>CURRENCY_CD_TABLE</publicationprocess>

            <publishtimestamp>2000-02-
25T16:01:19.000000</publishtimestamp>

            <status>4</status>

            <defaultdataversion>VERSION_1</defaultdataversion>

            <dataversions>

              <publicationdataversion>

                <version>VERSION_1</version>

                <data encoding="base64(deflate)"
encodedlength="536(401)" length="2346">your deflated and base64 encoded data
here</data>

              </publicationdataversion>

            </dataversions>

          </publication>

        </variable>

```

```

        </invoke>
    </operations>
</request>

```

4. The PeopleSoft application replies to the publish with an HTTP Response object:

```

<?xml version="1.0" ?>
<reply>
    <operations namespace="PublishSubscribe" interface="PublishSubscribeSystem">
        <invoke opnum="1" member="Publish">
            <return type="number">0</return>
            <variable type="object" interface="Publication">
                <publication>
                    <publishingnode>THIRD-PARTY</publishingnode>
                    <channel>ENTERPRISE_SETUP</channel>
                    <publicationid>38</publicationid>
                    <publishtimestamp>2000-02-25T16:02:52.203000-
0800</publishtimestamp>
                </publication>
            </variable>
        </invoke>
    </operations>
</reply>

```

PeopleSoft to Third Party XML/HTTP Conversation

A PeopleSoft node continues publishing XML messages to a remote node until it determines the remote node is down. Then, PeopleSoft node will ping the remote node with an HTTP Request, until the remote node responds to the ping.

Therefore, if the third party subscriber chooses to write their own subscription servlet (instead of writing a custom subscription handler for the PeopleSoft Application Gateway); then, they will need to write a response for the ping *and* for the message post.

1. PeopleSoft application sends a publish transaction with a new HTTP Request:


```

<?xml version="1.0" ?>

<request version="1.0">

  <from node="PEOPLESOFT" />

  <to node="THIRD-PARTY" />

  <operations namespace="PublishSubscribe" interface="PublishSubscribeSystem">

    <invoke member="Publish">

      <variable type="object" interface="Publication">

        <publication>

          <publishingnode>PEOPLESOFT</publishingnode>

          <channel>ENTERPRISE_SETUP</channel>

          <publicationid>3</publicationid>

          <subject>CURRENCY_SYNC</subject>

          <class>PSAPMSG</class>

          <originatingnode>PEOPLESOFT</originatingnode>

          <publisher>VP1</publisher>

          <publishtimestamp>2000-02-24T14:49:18.047000-
0800</publishtimestamp>

          <status>4</status>

          <defaultdataversion>VERSION_1</defaultdataversion>

          <dataversions>

            <publicationdataversion>

              <version>VERSION_1</version>

              <data encoding="base64(deflate)"
encodedlength="536(400)" length="2358">deflated and base64 encoded message body
will be here</data>

            </publicationdataversion>

          </dataversions>

        </publication>

      </variable>

    </invoke>

  </operations>

```

```
</request>
```

2. The third-party application replies with the HTTP Response object.



Note. The response header variable *content type* must be set to *text/xml*.

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

<reply>

  <operations namespace="PublishSubscribe" interface="PublishSubscribeSystem">

    <invoke member="Publish" opnum="1">

      <return type="number">0</return>

      <variable type="object" interface="Publication">

        <publication>

          <publishingnode>PEOPLESFT</publishingnode>

          <channel>ENTERPRISE_SETUP</channel>

          <publicationid>0</publicationid>

          <publishtimestamp>2000-02-25T13:33:20.453000-0800</publishtimestamp>

        </publication>

      </variable>

    </invoke>

  </operations>

</reply>
```

Ping and Response Sample

The following code shows a *ping* from PeopleSoft:

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

<request version="1.0">

  <from node="PEOPLESFT" />

  <to node="THIRD-PARTY" />

  <operations namespace="PublishSubscribe" interface="PublishSubscribeSystem">
```

```

    <invoke member="PingNode">

        <variable type="string">THIRD-PARTY</variable>

    </invoke>

</operations>

</request>

```

The third party application *replies* to the ping with the *Response* object



Note. The response header variable *content type* must be set to *text/xml*.

```

<?xml version="1.0"?>

<reply>

    <operations namespace="PublishSubscribe" interface="PublishSubscribeSystem">

        <invoke opnum="1" member="PingNode">

            <return type="number">0</return>

        </invoke>

    </operations>

</reply>

```



For more information, see the following Web sites:

<http://msdn.microsoft.com/xml/articles/beginner.asp>,

<http://msdn.microsoft.com/library/psdk/xmlsdk/xmlp6apg.htm>,

<http://www.amazon.com/exec/obidos/ASIN/1861001576/o/qid=951870261/sr=8-2/002-9946226-2205005>

Posting from a C Program

The following is an example of a C function that posts the XML document to the Web server for publication/subscription. It is *not* a product or utility supported by PeopleSoft, but an example of one way to do an HTTP post.

```

//***** PublishXmlMessageToLocation *****

EPUBCONSTATUS CPublicationContractManager::PublishXmlMessageToLocation(

```

```

    LPCTSTR szLocation, LPCTSTR szXml, int nXmlItems,

        int * lpnStatusCode)

{
#ifdef PS_WIN32

    EPUBCONSTATUS eStatus = PUBCONSTATUS_ERROR;

    // Parse the URL
    URL_COMPONENTSA url;

    memset(&url, 0, sizeof(URL_COMPONENTSA));

    url.dwStructSize = sizeof(URL_COMPONENTSA);

    // Indicate which components should be returned by setting the length to 1
    url.dwSchemeLength = 1;

    url.dwHostNameLength = 1;

    url.dwUserNameLength = 1;

    url.dwPasswordLength = 1;

    url.dwUrlPathLength = 1;

    char    szLocationAscii[MNDM_LOCATIONLEN + 1];

    _PStcstombs( szLocationAscii, szLocation, MNDM_LOCATIONLEN + 1 );

    // Parse the URL string
    if (!InternetCrackUrlA(szLocationAscii, strlen(szLocationAscii), 0,

        &url))

    {
        return eStatus;
    }

    if ((url.nScheme != INTERNET_SCHEME_HTTP)

```

```
        && (url.nScheme != INTERNET_SCHEME_HTTP))

    {

        return eStatus;

    }

char * pszHostName = new char[url.dwHostNameLength + 1];

strncpy(pszHostName, url.lpszHostName, url.dwHostNameLength);

pszHostName[url.dwHostNameLength] = 0;


if (url.nPort == 0)

    {

        if (url.nScheme == INTERNET_SCHEME_HTTPS)

            url.nPort = INTERNET_DEFAULT_HTTPS_PORT;

        else

            url.nPort = INTERNET_DEFAULT_HTTP_PORT;

    }


// convert XML to ASCII

char    *pszXmlAscii = new char[ nXmlItems + 1 ];

int iTemp = _tcslen( szXml );

_PStcstombs( pszXmlAscii, szXml, iTemp );

pszXmlAscii[iTemp] = '\\0';


HINTERNET hSession = NULL;

HINTERNET hConnection = NULL;

HINTERNET hRequest = NULL;

char szAcceptType[] = "text/xml";

char szContentType[] = "Content-Type: text/xml";
```

```
// Open HTTP session

hSession = InternetOpenA("Publication Contract Manager",

    PRE_CONFIG_INTERNET_ACCESS, NULL, NULL, 0);

// Open HTTP connection

hConnection = InternetConnectA(hSession, pszHostName, url.nPort, NULL, NULL,
    INTERNET_SERVICE_HTTP, 0, 1);

// free memory allocated for host name

delete[] pszHostName;

pszHostName = NULL;

int nRequestFlags = INTERNET_FLAG_EXISTING_CONNECT;

if (url.nScheme == INTERNET_SCHEME_HTTPS)

    nRequestFlags |= INTERNET_FLAG_SECURE;

// Open HTTP request

hRequest = HttpOpenRequestA(hConnection, "POST", url.lpszUrlPath,

    "HTTP/1.0", NULL, (const char **)&szAcceptType, nRequestFlags, 1);

// Send request

HttpSendRequestA(hRequest, szContentType, strlen(szContentType),

    pszXmlAscii, nXmlItems);

// Free xml buffer

delete [] pszXmlAscii;

pszXmlAscii = NULL;

HTTP_STATUS_CODE eStatusCode;
```

```
char szBuffer[80];

DWORD dwLen = sizeof(szBuffer); // / sizeof(TCHAR); = # of items

// Get status code

HttpQueryInfoA(hRequest, HTTP_QUERY_STATUS_CODE, szBuffer, &dwLen, NULL);

eStatusCode = (HTTP_STATUS_CODE)atol(szBuffer);

if ((int)eStatusCode == 0)

    *lpnStatusCode = 404;

else

    *lpnStatusCode = (int) eStatusCode;

if (eStatusCode == 0)

{

    // checking for no return code - failed InternetConnect

    eStatus = PUBCONSTATUS_ERROR;

}

else if (eStatusCode < _200_OK)

{

    // 100-199: Informational

    eStatus = PUBCONSTATUS_RETRY;

}

else if (eStatusCode < _300_Multiple_Choices)

{

    // 200-299: Successful

    eStatus = PUBCONSTATUS_DONE;

}

else if (eStatusCode < _400_Bad_Request)
```

```
    {  
        // 300-399: Redirection  
    }  
else if (eStatusCode < _500_Internal_Server_Error)  
    {  
        // 400-499: Client Error  
        eStatus = PUBCONSTATUS_ERROR;  
    }  
else  
    {  
        // 500-599: Server Error  
        eStatus = PUBCONSTATUS_RETRY;  
    }  
  
    // Close handles  
    InternetCloseHandle(hRequest);  
    InternetCloseHandle(hConnection);  
    InternetCloseHandle(hSession);  
  
    return (eStatus);  
  
#endif    //PS_WIN32
```


CHAPTER 7

Application Messaging Architecture

This section contains information related to the PeopleSoft Application Messaging architecture. Our messaging solution adheres to the publish/subscribe model. At least in terms of your PeopleSoft applications, you can use the terms messaging and publish/subscribe interchangeably.

Before you get started configuring your Subscription Server Handlers or the Application Messaging Gateway, we encourage you to read the information in this book and the following chapter to become familiar with all of the components involved as well as the utilities we offer for configuration. Also, you should become familiar with the information regarding the development side of application messaging.



For more information on the development steps required for application messaging, see *Designing Application Messages*.

First, it's important to recognize that configuring your application messaging server components is really only half of the job. The other half of the job lies on the development side. Typically, system administrators, such as DBAs, are responsible for setting up the machinery involved in your enterprise information system. Working in a separate but coordinated capacity there is usually a development staff whose main focus is developing or customizing PeopleSoft applications to deploy across the enterprise.

As you read the following topics, you will arrive at the conclusion that setting up the PeopleSoft Application Messaging solution requires detailed coordination between the system administration staff and the development staff. For instance, you use Application Designer to define Messages, Message Channels, and Message Nodes. Then you use PSADMIN to configure the nodes and server processes on the server side.



Note. Theoretically, one team could handle both defining and configuring the messaging system, however, the development side requires knowledge of Application Designer, PeopleCode, and general familiarity with the Peoplesoft development environment. In most cases, the development staff is separate from the system administration staff.

After reading this chapter you will be familiar with the following topics:

- The purpose of PeopleSoft Application Messaging.
- The terminology associated with all of the messaging components.
- The architecture and components involved in our messaging solution.

Why Use Application Messaging?

If you had a single set of applications running against a single database and you ran your entire business from the information stored in the single database, you probably won't need an application messaging or publish/subscribe system. However, many organizations share data between different information systems internally, and with the *external* information systems of those of their customers and partners.

Suppose at your site some employees' time card information is stored in a Microsoft Access database, which when payroll runs needs to be reflected in a PeopleSoft database where you have PeopleSoft Payroll. Let's also say that each of these databases needs to be synchronized with your PeopleSoft HR database that may be on a different site. For example, each time a pay rate, employee classification, or department value changes, all these databases need to reflect the current values to keep the system in sync and maintain the integrity of your data.

Here is a simple example where PeopleSoft Application Messaging can be used.

Application Messaging provides the capability for synchronizing data from one application system to another. Using the example, if you changed the department to which an employee belonged in your main HR database, the HR application could publish the new department information. The timecard and payroll applications and databases could then subscribe to publications generated from the HR database. Then having subscribed to this type of publication the appropriate other systems would be notified of the change and update any data as needed.

The publish-and-subscribe mechanism provides integration across PeopleSoft applications and product lines, as well as with third-party applications that is close to real-time.

Application messaging provides many benefits over previous solutions for system integration. However, the two key benefits are as follows:

- Synchronization of data
- System-to-system workflow

There are numerous integration scenarios in which application messaging is advantageous:

- Application-to-application integration.
- System-to-system integration.
- Cross-release integration.
- Business-to-business integration.

Terminology

The following list of terms is designed to help you become familiar with the components as well as the vocabulary involved the PeopleSoft Messaging Architecture. We suggest that you take a moment to review the following list of terms before you move on to the following sections.

Message

Defines the record(s)/fields to be published/subscribed. If you use a message for subscription, then it defines the subscription process as well. Generated in XML format.

XML

A standard protocol used for publishing and subscribing data between nodes. The syntax is very basic and similar to HTML scripting in that it uses tags to identify the assignment of variables, fields, types, and so on. For example,

```
<EMPLOYEE_NAME>Sawyer, Greg</EMPLOYEE_NAME>
```

Message Queue

The database tables that store all messages.

Publish

To populate and send a pre-defined application message as part of processing a transaction.

Subscribe

To indicate interest in particular pre-defined application messages to be processed with a subscription routine executed with PeopleCode or Application Engine.

Publication

An instance of an application message stored in the message queue.

Publication Contract

The Publication Broker Dispatcher creates one publication contract for each remote node to which the publication needs to be routed. Then the Publication Broker Handler fills the publication contract by delivering the message.

Subscription Contract

When the Subscription Broker Dispatcher receives a published message, it creates a subscription contract for the Subscription Broker Handler. The handler then fills the contract by making sure the subscription PeopleCode executes.

Application Messaging Gateway

The gateway is a collection of servlets that enable you to administer and view information regarding the message nodes in your system, and, most importantly, it is the component that routes messages to PeopleSoft and non-PeopleSoft nodes. The collection of servlets consists of the reader servlet for viewing node information, the configuration servlet administering nodes, and the gateway servlet for delivering messages throughout your system.

Message Node (Node)

A message node consists of a PeopleSoft database and at least one application server, or a message node could be a third party system. Nodes publish messages to other remote nodes by way of the Application Messaging Gateway.

Message Channel

Associates messages to routing rules. Defines how a transaction is to be published or subscribed to. You can configure channels to define "parallelism" and order of processing for the messages (also known as creating sub-channels).

PeopleSoft Messaging Architecture

Before you begin configuring your messaging system, we suggest that take the time to become familiar with all of the components within the PeopleSoft messaging architecture and see how they work together to allow messages to be published throughout your enterprise. After reading this section you'll have a better understanding of nodes, how the web server is used, and how the server processes handle publications and subscriptions.

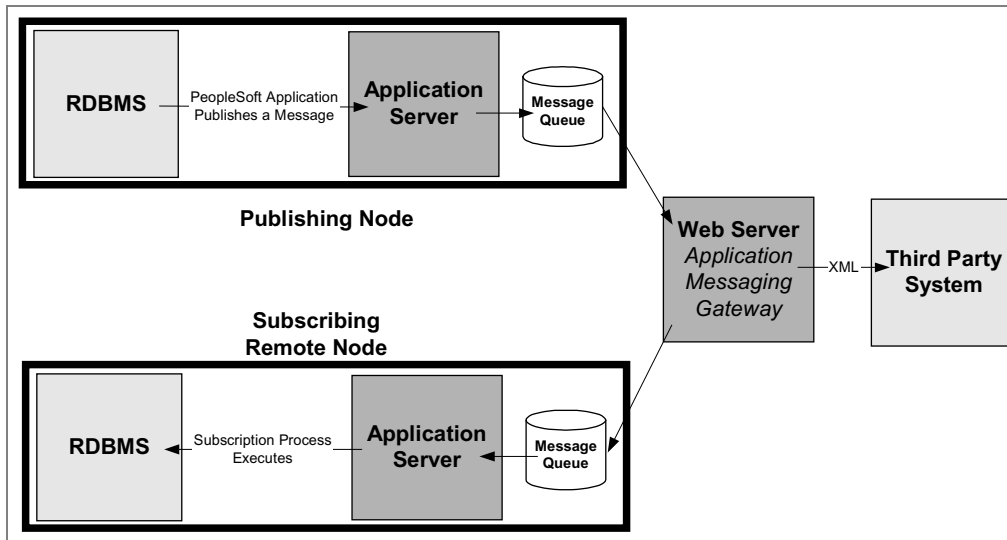
Required Components

There are a variety of tasks involved in setting up your application messaging system. Before you can publish your first message and actually have it be received by a remote node, you need to complete all of the development tasks within Application Designer, as well as configure all of the server-side components to support the Application Designer definitions. In this section we discuss the server-side components.

The PeopleSoft Application Messaging architecture requires the following components:

- A local node defined on the publication system.
- At least one PeopleSoft message node representing the subscribing system comprised of a PeopleSoft database and application server.
- A supported Web Server to house the Application Messaging Gateway.

The following example demonstrates how each of the components fits into the overall messaging functionality. Assume that the publishing node is Human Resources database on which a user just updated an address in the personal information tables. When the user saves the panel, for example, that event might invoke message publication that the application server routes to the Application Messaging Gateway. Let's say that the subscribing remote node is defined to receive messages published that carry a change to personal information records. As a result, the Application Messaging Gateway routes the message to the subscribing remote node (and any other subscribing nodes). If the message needs to be consumed by a third party system, the Application Messaging Gateway can also expose a message for external systems.



High-Level Messaging Architecture



Note. The Message Servers runs on the application server, while the Message Queue is a set of database tables.

Notice that the delivery of the message to the remote node occurs by way of the Application Messaging Gateway. If a message is to be delivered to a third party system, make note that the Application Messaging Gateway does not deliver the message directly. PeopleSoft delivers functionality to write the message to an XML file for your custom program to consume.

Nodes

What exactly is a node? In the simplest terms, a PeopleSoft node is the combination of a database and at least one application server. A node can also be defined as an external system that is either the source or destination of a message. In short, a node is one of the end points of an Application Messaging System.

The messaging definitions that you created in Application Designer exist within the database, and the application server allows the publications and subscriptions to link it to message channels by way of the Application Messaging Gateway.

The database acts as the core of each PeopleSoft node. Since each set of PeopleSoft applications has its own database, it's typical to see PeopleSoft nodes named after a particular PeopleSoft application. For instance, you might have an HR node, and EPM node, and so on.

When discussing message nodes, it's important to consider the attribute of local and remote nodes. Whether a node is local or remote is determined by which database a node is defined in. If you are connected to database X defining node X, then node A is local. But later, if you are connected to node Y, then node X is remote. Local and remote are therefore relative terms.

The following sections explore these concepts further.

Local

A local node refers to the node that publishes a message. A published message originates from one local node, and that message can fill subscriptions to one or more remote nodes.

A local node can publish a message to itself, and this would apply to the following situations:

- In many instances, different sub sets of applications within the same product line use different tables to store the same data within a database. For example, in your HR database, PeopleSoft Payroll and the core HR applications share data, but store the shared data in different tables. If you were to make a change to an employee's main HR data, you could publish a message to the local node to make sure that the Payroll component running on the same database is notified and updated. This allows you to maintain almost instantaneous data synchronicity.
- Even if a message is intended for a remote node, the system can still publish a message locally. Then after the publication occurs, whatever routing rules you have defined for the message take over and ensure that the message gets published to any remote nodes as defined in your routing rules.



Note. Most likely, the majority of messages will be published to remote PeopleSoft nodes or third party nodes.

Remote

A remote node refers to a distinct node that subscribes to a message published by another node. By distinct, we mean a node that is separate from the publishing (or local) node. There are two kinds of remote nodes: a remote PeopleSoft node, and a remote non-PeopleSoft node.

- **Remote PeopleSoft Node.** This is a node, separate from the local node, that contains a PeopleSoft application server and a PeopleSoft database running at least PeopleTools 8.0 software. A system can have *n* number of PeopleSoft nodes, yet each one needs to be registered on the Lookup table in the PeopleSoft Gateway Servlet in the Application Messaging Gateway.
- **Remote Non-PeopleSoft Node.** A remote node can also be a non-PeopleSoft, or third party, node, such as an Oracle or SAP system. If you publish to a non-PeopleSoft node, the message will also be sent through the Application Messaging Gateway. However, a different servlet processes the message: the External Gateway Servlet.



Note. Keep in mind that if you are publishing a message to a PeopleSoft node containing a PeopleSoft database running on a version of PeopleTools prior to PeopleTools 8.0, then technically, you are publishing to a non-PeopleSoft node.



For more information on configuring the Application Messaging Gateway see *Administering the Application Messaging Gateway*.

Application Server

The application server is the heart of the PeopleSoft architecture, and with each successive release it becomes even more essential to our new technology. The application messaging solution is no exception. You *can not* configure our messaging solution without first installing and configuring a PeopleTools 8.x application server.



For more information on installing the PeopleSoft Application Server see the PeopleSoft *Installation and Administration* books.

Messaging Server Processes

Just as you have other separate server processes on the application server, such as PSAPPSRV and PSQCKSRV, designed to handle specific types of transaction requests, the application server offers six server processes related specifically to application messaging. They are:

- PSBRKDSP. Broker Dispatcher.
- PSBRKHND. Broker Handler.
- PSSUBDSP. Subscription Dispatcher.
- PSSUBHND. Subscription Handler.
- PSPUBDSP. Publication Dispatcher.
- PSPUBHND. Publication Handler.

When you boot your application server, these server processes will start automatically if you've opted to configure the messaging servers. Out of the box, PeopleSoft provides a default, or demo, version of each of these server processes. The default server processes have a _df1t at the end. You can customize the configuration settings for each one just as you would for PSAPPSRV or PSQCKSRV.



For more information on specify tuned variables for the application messaging server processes, see Messaging Server Administration.

These server processes operate in sets of sixes, and within that set of six there are three sets of two server processes that work as a unit. For instance, you can't configure PSBRKDSP and PSBRKHND to operate independently of the other. From an administrative perspective, these server processes can only be configured in the assigned pairs.

These three sets of two server processes are very important to the mechanics of publishing and subscribing to messages. Each assigned pair comprises one of the following messaging components: the Publication Broker, the Publication Contractor, or the Subscription Contractor. The pairs of server processes are divided and assigned accordingly:

- Publication Broker
 - PSBRKDSP
 - PSBRKHND
- Publication Contractor
 - PSPUBDSP
 - PSPUBHND
- Subscription Contractor
 - PSSUBDSP
 - PSSUBHND



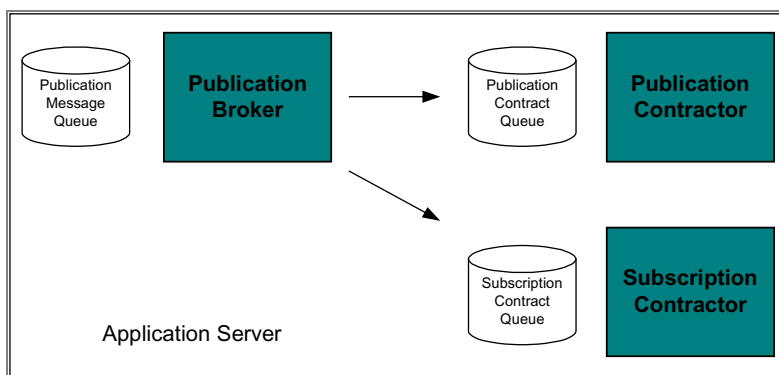
Note. To improve performance of a busy messaging system you can add a set of dedicated messaging server processes to handle the messaging workload of a particular message channel.



For more information on setting up dedicated messaging server processes, see *Messaging Server Administration*. For more information on Message Channels, see *Defining Message Channels*.

Brokers and Contractors

The Publication Broker, Publication Contractor, and Subscription Contractor are the primary application server components required for application messaging. In terms of a hierarchy, the Publication Contractor and the Subscription Contractor are on the same level below the Publication Broker, which distributes, or routes, the workload to both Contractor server processes.



Brokers and Contractors

Notice that each component has its own queue to hold the messages it needs to process. The following sections describe what each component does.

Publication Broker

The Publication Broker acts as the routing mechanism. When a message arrives in its queue, the Broker runs the defined routing rules. If the message needs to be published to a remote node it routes the message to the Publication Contractor. On the other hand, if the message is subscribed to on the local node, then the Broker routes the message to the Subscription Contractor.

Routing involves submitting either a Subscription or Publication Contract to the appropriate contractor followed by an asynchronous call to the contractor notifying it that it has work in its queue.

Publication Contractor

The Publication Contractor references the Publication Contract submitted by the Broker, and performs an HTTP post of the publication message to the Application Messaging Gateway.

When the Application Messaging Gateway sends a reply indicating that it received the publication message, the Publication Contractor updates the Publication Contract with the status of subscription processing (done or retry).

The dispatcher and handler components perform all of this processing: PSPUBDSP and PSPUBHND.

Subscription Contractor

The Subscription Contractor references the Subscription Contract submitted by the Publication Broker, and it executes the appropriate subscription PeopleCode. Then it updates the Subscription Contract concerning the status of the subscription processing.

As with the Publication Contractor, a dispatcher and a handler component perform all of this processing: PSSUBDSP and PSSUBHND.

Dispatchers and Handlers

As mentioned previously, the Publication Broker, Publication Contractor, and Subscription Contractor, are comprised of two separate server processes working together to handle any 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/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, you have one listener to 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, think of the dispatcher as the listener, and the handler as similar to the JSH.

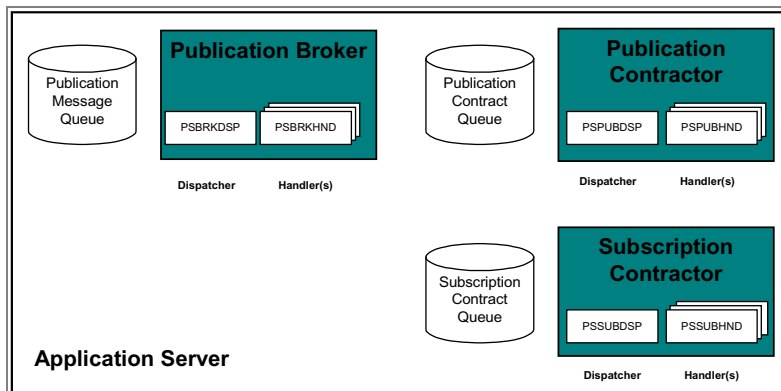
So, for the components discussed thus far (Publication Contractor, Subscription Contractor, and Publication Broker) there are *at least* two server processes doing the work: a single “dispatcher” and one or more “handlers”. The PSXXXDSP server process is the dispatcher, and the PSXXXHND server process 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 for the Publication Broker.

The following example depicts the messaging server processes grouped within their specific functional position within the messaging architecture.



Dispatchers and Handlers

Web Server

The web server fulfills a critical junction within the application messaging system. The only way to publish a message to a remote PeopleSoft node is by way of the Application Messaging Gateway (also called the gateway, running on a supported Web server. The Application Messaging Gateway consists a collection of servlets running on the Web server. One servlet is designed to transmit messages to message nodes, while the others are used for viewing and administering the node information.

The gateway consists of PeopleSoft files that you install and configure to your web server. You configure handlers for delivering messages. A PeopleSoft handler delivers all messages to PeopleSoft nodes, and there is another handler that PeopleSoft delivers for exposing messages to third party systems. If needed, you can also develop your own custom handlers for additional third party messaging solutions.

The PeopleSoft Gateway Servlet allows you to manage a table, called the Lookup Table, which acts as a directory for all the nodes that are part of the PeopleSoft messaging system. A proper listing in the Lookup Table allows all applicable messages to be routed to the appropriate, subscribing nodes located on a particular application server/database system.

When the appropriate node appears in the Lookup Table, the PeopleSoft handler, using Jolt, sends the publication message to the PSAPPSRV in the target node using the connect information found for that node in the Lookup Table. The PSAPPSRV then creates a publication message that will be handled *locally* within the target node.

To deliver messages to non-PeopleSoft nodes, we obviously can't just send a Jolt message to the application server since a non-PeopleSoft node will not have a PeopleSoft Application Server running against its database.

Consequently, we have identified the web server as a third party integration point and developed a handler. PeopleSoft can expose the entire contents of the message, however, it's up to your intended third party system to provide the appropriate API. Most industry standard software systems include such APIs. PeopleSoft provides source Java code for you to customize the servlet to suit your transaction requirements.



For more information on configuring the Application Messaging Gateway servlets see *Administering the Application Messaging Gateway*.

Sample Message Journey

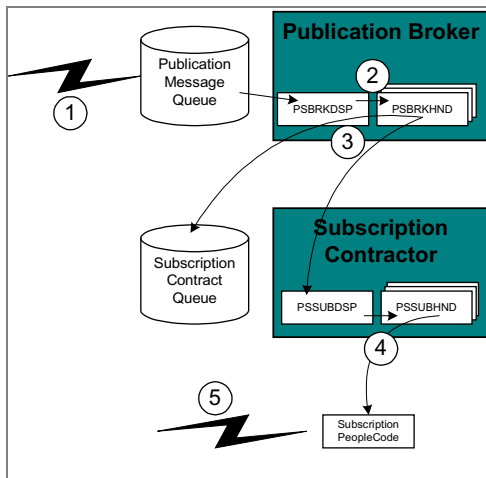
The previous sections introduced all of the components involved in the PeopleSoft Messaging architecture. However, to truly gain an understanding and appreciation of each component's role within the system, it's useful to describe, step-by-step, the path a message travels during a publish/subscribe transaction.

The following sections present the possible publication scenarios and describe the events that occur at each step. After reading this section you will have an understanding of each of the possible publish/subscribe scenarios: Local Node, Remote PeopleSoft Node, and non-PeopleSoft Node.

The descriptions appear in the form of an illustration accompanied by a numbered sequence of events.

Publishing to a Local Node

The following example depicts the sequence of events that occur within the messaging system during a local node publish.



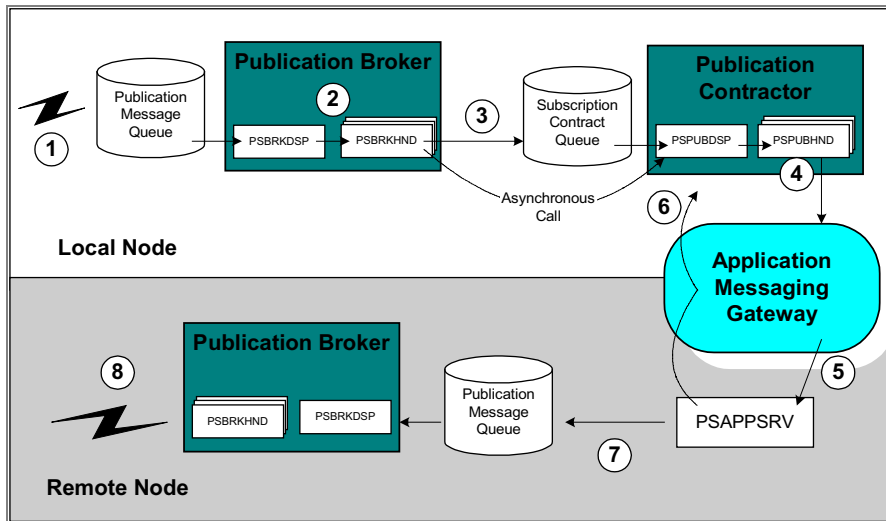
Local Node Publish

The following table describes the events that occur at each step in the publication transaction to a local node.

Step	Description
1	Publication message arrives in the message queue.
2	The Publication Broker dispatcher (PSBRKDSP) routes the message to an available handler (PSBRKHND), which reads the data in the publication message. The handler then runs the Publication routing rules to see where publication is to be delivered, and it runs the Subscription routing rules to see if the message is to be processed locally.
3	For this scenario, the publication message indicates local publication. So, the Publication Broker handler submits a Subscription contract to the Subscription Contractor's queue, and it also sends an asynchronous call to notify the Subscription Contractor that an item is waiting in its message queue.
4	The Subscription Contractor dispatcher reads the data in the Subscription contract and notifies its handler to run the appropriate subscription PeopleCode while updating the Subscription Contract with the status of the subscription processing.
5	The Subscription PeopleCode updates the appropriate record in the local database.

Publishing to a Remote PeopleSoft Node

The following example depicts the sequence of events that occur within the messaging system during a remote node publish.



Publishing to a Remote PeopleSoft Node

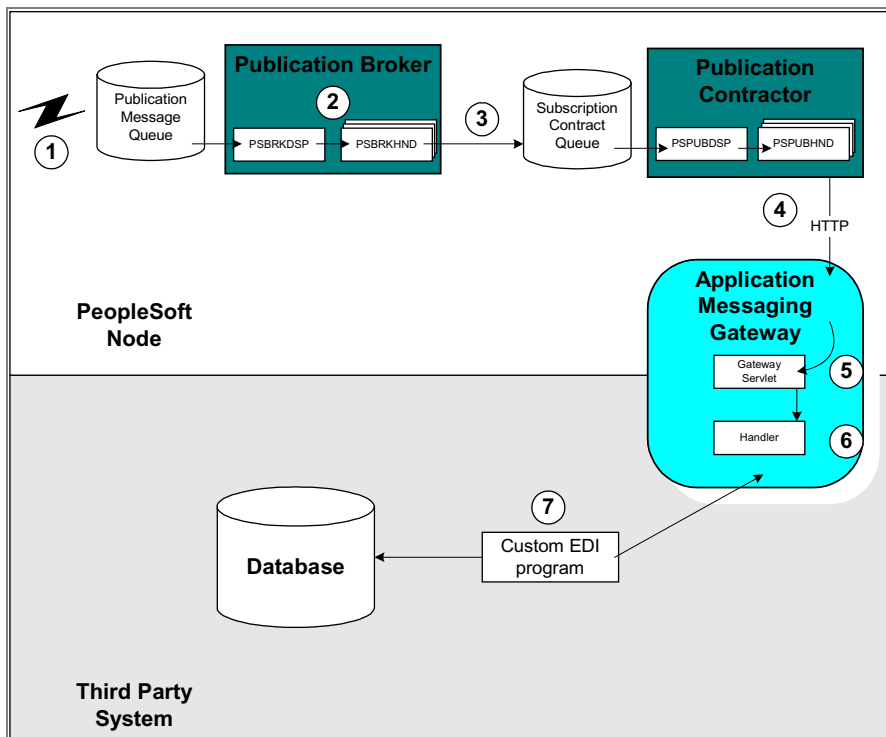
The following table describes the events that occur at each step in the publication transaction for a remote PeopleSoft Node.

Step	Description
1	The Publication message arrives into the message queue.
2	The Publication Broker dispatcher routes the message to an available handler, which reads the data in the publication message. The handler then runs the Publication routing rules to see where publication is to be delivered, and it runs the Subscription routing rules to see if the message is to be processed locally.
3	For this scenario, the publication message indicates remote publication. So, the Publication Broker handler submits a Publication Contract to the Publication Contractor to handle. Simultaneously, the Publication Broker sends an asynchronous call to notify the Publication Contractor that an item that needs to be processed currently waits in its message queue.
4	The Publication Contractor reads the data in the Publication Contract and notifies its handler to perform an HTTP post of the publication to the Application Message Gateway. When it receives a reply from the Application Message Gateway indicating acceptance of the publication message, the Publication Contractor updates the Publication Contract with the status of the subscription processing.
5	After receiving the message from the Publication Contractor, the Application Messaging Gateway sends a publication message to the remote node. The gateway servlet looks up the node in the Lookup table and finds the appropriate connect information necessary to invoke a service on the subscribing application server/database. The gateway retrieves the information using a synchronous call, by way of Jolt, to the PSAPPSRV server process on the application server in the target node.
6	PSAPPSRV, by way of the Application Message Gateway, sends a message to the originating node indicating that it received the publication message. PSAPPSRV then constructs and sends an XML reply indicating whether the publication

	message was processed successfully.
7	PSAPPSRV then creates a publication message that will be processed locally on the “remote” node.
8	The Publication Broker on the remote node, after receiving the publication message, publishes the message locally just as described in the Local Node publication. See Local Node for this description.

Publishing to a Third Party System

The following example depicts the sequence of events that occur within the messaging system during a publication to a non-PeopleSoft node.



Publishing to a Third Party System

The following table describes the events that occur at each step in the publication transaction to a non-PeopleSoft Node.

Step	Description
1	The Publication message arrives into the message queue.
2	The Publication Broker dispatcher routes the message to an available handler, which reads the data in the publication message. The handler then runs the

	Publication routing rules to see where publication is to be delivered, and it runs the Subscription routing rules to see if the message is to be processed locally.
3	For this scenario, the publication message indicates remote publication. So, the Publication Broker handler submits a Publication Contract for the Publication Contractor to handle and sends an asynchronous call to notify the Publication Contractor that an item is waiting in its message queue.
4	The Publication Contractor reads the data in the Publication Contract and notifies its handler to perform an HTTP post of the publication to the Application Messaging Gateway. When it receives a reply from the Application Messaging Gateway indicating acceptance of the publication message, the Publication Contractor updates the Publication Contract with the status of the subscription processing.
5	The system hands the publication to the gateway servlet to process. Given the list of registered handlers that is available in the servlet process, the servlet searches through each handler to find which handler has this particular remote node defined. Once found, it asks the handler if it's interested in the entire publication or just the data portion of the message. The servlet then hands the appropriate data, for the most part in XML form, by way of a Java API call, for the handler to consume.
6	The handler consumes the publication and returns a status code back to the external gateway servlet. Based on this status code, the servlet constructs the appropriate reply document and sends it back to the publishing system.

CHAPTER 8

Messaging Server Administration

Although the server processes devoted to your messaging system are all part of the larger application server domain, they do comprise a distinct set of server processes that are not involved with the ordinary transactions associated with PIA connections. You can think of the six server processes required for transmitting messages throughout your messaging system as a separate sub-component of your application server domain; this sub-component is the messaging server.

To help distinguish the separate identity of your messaging server, we have included a separate menu for administering your messaging server processes. This menu is the Messaging Server Administration menu. You select this menu from the main PeopleSoft Domain Administration menu.

```
-----  
PeopleSoft Domain Administration  
-----  
  
Domain Name: PS800  
  
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) :

From this menu you can add new messaging servers, edit the channel list for messaging servers, and delete any messaging servers that are no longer needed.



Note. Even though you add new messaging servers using this separate menu, you still configure the messaging server processes with PSADMIN just as you would any other server process.

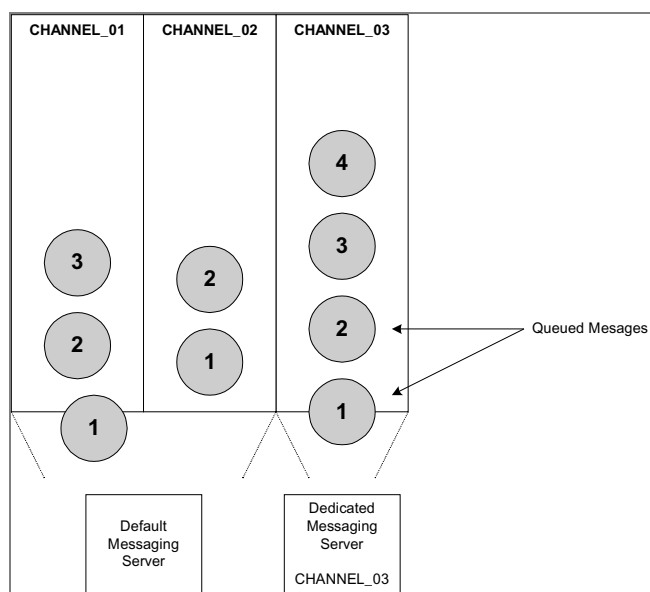
The following sections explain how to create additional messaging servers and edit their channel lists. And in doing so, we'll also explain the advantages of adding more messaging servers and how the channel list becomes important.

Adding Dedicated Message Servers

PSADMIN offers you a set of default message server processes that comprise the default, or demonstration, messaging server. The demonstration messaging server is sufficient for development, testing, or demonstrations, but, in most cases, you'll add multiple messaging servers to increase the performance of your messaging system.

As the volume of published messages increases in a production system, it's likely that one messaging server could become overloaded. To avoid any potential overloads and performance degradation, PeopleSoft suggests that you add additional messaging servers to cope with an increase in message volumes.

Not only can you add more messaging servers, but you can specify particular message channels to which they are solely assigned. You assign a messaging server to a particular channel using the channel list. So if a given channel is notoriously the most active and creating performance bottlenecks, you can assign the appropriate number of messaging servers to that channel to cope with the message volume. A messaging server is capable of handling multiple message channels depending on your configuration. When you assign a messaging server to a particular channel, you are implementing a *dedicated* server.



Dedicated Messaging Server

The previous example depicts a dedicated messaging server assigned to CHANNEL_03. In this scenario, the default messaging server (_dflt process collection) continues to process the messages in the other channels while the dedicated messaging server only processes the messages within a specified Message channel. Unless you create and configure dedicated messaging servers, the default messaging server handles all incoming messages. Remember that a messaging server is the collection of the six messaging server processes.



Note. Before you can assign messaging server processes to message channels, you need to *first* define the message channels using Application Designer.



For more information on defining message channels, see [Defining Message Channels](#).

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

- **Add the new messaging server.** You add a new messaging server using 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 particular message channels.
- **Configure the new messaging server.** By adding a new messaging server of any type, you will notice that your configuration files get updated to include parameters for the new server processes. Since a messaging server consists of six server processes, when you create a new one, you will see six additional configuration sections in PSADMIN's domain configuration menu. They'll look identical to the _dflt messaging server processes except they'll reveal the name you gave them in place of the "_dflt." In short, in order for any new messaging server processes to take effect, you must first reconfigure your domain to include the new parameters.



Note. Typically, you'll be adding multiple messaging components at a time, so PeopleSoft suggests that you add all of the messaging components, and then reconfigure the domain all at once.

After reading the following sections, you will be familiar with the procedures you'll need to complete for creating new messaging servers, editing channel lists, and removing unneeded messaging servers.

Creating and Assigning Dedicated Servers

The following procedure describes the steps involved in creating and assigning a dedicated messaging server to selected Message channels. Keep in mind that a messaging server consists of six server processes grouped as by twos as the Publication Broker, Publication Contractor, and the Subscription Contractor. This means that when you create a dedicated messaging server, you actually need to complete three procedures. One creates the Publications Broker, while the remaining two create the Publication Contractor and Subscription Contractor.

When adding dedicated messaging servers keep the following guidelines in mind:

- You will be prompted to name the server processes. For consistency sake, you should devise some naming guidelines.
- There is no logic on the application server that checks to see if the channel to which you assign a dedicated messaging server actually exists. For that reason, you need to check with development to make sure it exists and that you have the appropriate name.
- A dedicated messaging server needs to be assigned to at least one channel.

To create a dedicated Publication Broker

1. From the PeopleSoft Domain Administration menu, select **Messaging Server Administration menu**.

```

-----
PeopleSoft Domain Administration
-----

Domain Name: PS800

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) : 7

2. From the Messaging Server Administration menu, select **Create a new messaging server**.

```

-----
Messaging Server Administration menu
-----

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



Note. If you have not previously dedicated any messaging servers you'll see the << *No dedicated messaging servers are defined* >> message.

3. Enter 1 for Publication broker

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): 1

4. Enter a name to identify the new Publication broker.

Enter name of the new server (alphanumeric, 6 char max) : **PT8MSG**

You are limited to a 6-character name. The name that you add will be appended to the general server process name. In this case, the name would appear as PSBRKSRV_PT8MSG.

5. Specify the message channels to which you will assign the newly added messaging server.

Enter the messaging channels to be handled by server 'PT8MSG': **HRMS_01**



Note. The message channel you specify should already be defined in Application Designer. The channel name that you enter must match *exactly* the name that appears in Application Designer. There is no prompt or validation that occurs between PSADMIN and the Application Designer definitions.

After you press enter, you should see the following on the screen:

```
Writing new CFG...Done.
```

```
Writing new UBX...Done.
```

Now that you have configured at least one dedicated server process, the Messaging Server Administration menu now appears as shown:

```
-----
                        Messaging Server Administration menu
-----
```

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

SERVER NAME	TYPE	CHANNELS
-----	----	-----
PT8MSG	BRK	HRMS_01

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) :

To add a dedicated Publication Contractor

1. From the PeopleSoft Domain Administration menu, select Messaging Server Administration menu.

```

-----
PeopleSoft Domain Administration
-----

      Domain Name: PS800

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) : 7

2. From the Messaging Server Administration menu, select **Create a new messaging server**.

```

-----
      Messaging Server Administration menu
-----

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

```

SERVER NAME	TYPE	CHANNELS
PT8MSG	BRK	HRMS_01

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

3. Select **Publication contractor** from the following menu.

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): 2

4. Enter a name to uniquely identify the Publication Contractor server process.

Enter name of the new server (alphanumeric, 6 char max) : **PCNT01**



Note. The name you select can not share the same name that you have chosen for any other messaging server.

5. Specify the Message channels that this Publication Contractor will service.

Enter the messaging channels to be handled by server 'PCNT01': HRMS_01

After pressing ENTER, you should see the following on the screen:

Writing new CFG...Done.

Writing new UBX...Done.

And the Messaging Server Administration menu should appear as shown:

```
-----
                        Messaging Server Administration menu
-----
```

In addition to the default messaging servers, the following

dedicated messaging servers are in the domain configuration:

SERVER NAME	TYPE	CHANNELS
-----	----	-----
PT8MSG	BRK	HRMS_01
PCNT01	PUB	HRMS_01

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) :

To add a dedicated Subscription Contractor

1. From the PeopleSoft Domain Administration menu, select **Messaging Server Administration menu**.

PeopleSoft Domain Administration

Domain Name: PS800

- 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) : 7

2. From the Messaging Server Administration menu, select **Create a new messaging server**.

Messaging Server Administration menu

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

SERVER NAME	TYPE	CHANNELS
-----	----	-----
PT8MSG	BRK	HRMS_01

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

3. Select **Publication contractor** from the following menu.

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): **3**

4. Enter a name to uniquely identify the Publication Contractor server process.

Enter name of the new server (alphanumeric, 6 char max) :**SCNT01**



Note. The name you select can not share the same name that you have chosen for any other messaging server.

5. Specify the message channels that this Publication Contractor will service.

Enter the messaging channels to be handled by server 'SCNT01': **HRMS_01**

After pressing ENTER, you should see the following on the screen:

Writing new CFG...Done.

Writing new UBX...Done.

And the Messaging Server Administration menu should appear as shown:

```
-----
                        Messaging Server Administration menu
-----
```

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

SERVER NAME	TYPE	CHANNELS
PT8MSG	BRK	HRMS_01
PCNT01	PUB	HRMS_01
SCNT01	SUB	HRMS_01

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) :

Configuring Dedicated servers

As stated previously, simply adding a new messaging server is only half of the process. The other half consists of reconfiguring the domain using PSADMIN.

Just as you configure any other server process that runs in an application server domain, you must also configure the server process to run with any custom parameter values. At the very least you need to reconfigure the domain so that it includes the new server process when it boots.



For more information on the parameters associated with messaging servers, see Messaging Server Administration.

Editing Message Server Channel Lists

After you've created a Publication Broker, Publication Contractor, or Subscription Contractor, you may need to add more message channels to the channel list, or you may want to decrease the number of message channels it services to improve performance.

At any rate, in order to change the channel list of an existing message server complete the following procedure.

To modify a channel list

1. From the PeopleSoft Domain Administration menu, select **Messaging Server Administration menu.**

```

-----
PeopleSoft Domain Administration
-----

Domain Name: PS800
  
```

- 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) : **7**

- 2. From the Messaging Server Administration menu select Edit the channel list for a messaging server.**

```
-----
                        Messaging Server Administration menu
-----
```

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

SERVER NAME	TYPE	CHANNELS
PT8MSG	BRK	HRMS_01
PCNT01	PUB	HRMS_01
SCNT01	SUB	HRMS_01

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

- From the list of defined servers, select the message server for which you want to modify the channel list.

	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

- Modify the channel list.

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

After pressing ENTER you should the following on the screen:

Writing new CFG...Done.

Writing new UBX...Done.



Note. You need to *explicitly* include any previous channels that appear in the **Current channels:** brackets to maintain their spot in the channel list. That's how you delete a channel from the channel list: by not including it in the new list that you enter at the **Enter new channels:** prompt. For instance, if you wanted to remove HRMS_01 from the channel list, you would just enter HRMS_02.

Removing Unneeded Message Servers

There will be times when a message server that you've previously created is no longer needed. Rather than have it boot and consume valuable system resources, it's best to just remove it from your domain. To do so, complete the following procedure.

To remove message servers from a domain

1. From the PeopleSoft Domain Administration menu, select **Messaging Server Administration menu**.

```
-----
PeopleSoft Domain Administration
-----

Domain Name: PS800

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) : 7

2. From the Messaging Server Administration menu, select **Delete an existing messaging server**.

```
-----
Messaging Server Administration menu
-----

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

SERVER NAME	TYPE	CHANNELS
-----	----	-----
PT8MSG	BRK	HRMS_01
PCNT01	PUB	HRMS_01
SCNT01	SUB	HRMS_01
test	PUB	test

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) : 3

3. From the server list, select the messaging server you want to remove from the domain.

	SERVER NAME	TYPE	CHANNELS
	-----	----	-----
1)	PT8MSG	BRK	HRMS_01
2)	PCNT01	PUB	HRMS_01
3)	SCNT01	SUB	HRMS_01
4)	test	PUB	test

Enter the number of the server to be deleted: 4

You should see a message looking similar to the following:

Server 'test' deleted.

Writing new CFG...Done.

Writing new UBX...Done.

Configuring Messaging Servers in PSADMIN

As mentioned previously, after you add a set of dedicated messaging servers, you then need to configure them so they boot when you start the application server. You configure your messaging servers using PSADMIN just as you do any other server process that runs on the application server. Before you attempt to configure additional messaging server processes, you should be very familiar with the other server processes that run on the application server.



For more information on the application server see [Connecting through PIA](#).

As stated in the previous chapters, two separate server processes comprise broker server, publication server, and subscription server. These two processes are a dispatcher and a handler.

When you are configuring a dispatcher server process you have different parameters to set than when you are configuring a handler process. The following sections describe the parameters you set for each type of process. Most of the parameters are very similar to other server processes, such as PSSAPPSRV, but there are a few specific parameters for messaging servers.



Note. The following sections also apply to the `_dflt` messaging server processes. Only one parameter is different between a dedicated messaging server process and its `_dflt` counterpart, and that is the parameter allowing you to add message channels to the channel list. Specifically, this is the `Channels=` parameter. The `_dflt` server processes can not be associated with any specific message channel.

Dispatcher Parameters

The following topics describe the parameters that apply to the dispatcher process which include:

- PSBRKDSP
- PSPUBDSP
- PSSUBDSP

Recycle Count

Specifies the number of times each server process will be executed before being terminated (intentionally) by PeopleSoft and then immediately restarted. Servers must be intermittently recycled to clear buffer areas. The time required to recycle a server is negligible—occurring in milliseconds. Recycle Count does not translate into a native Tuxedo parameter in the PSAPPSRV.UBB file. Instead the value is stored in memory and is managed by PeopleSoft.

Allowed Consec Service Failures

This option allows for dynamic server process restarts in the event of service failures. To enable this option, enter a number greater than zero, and to disable this option enter 0. The default for this parameter is 2. The numerical value you enter is the number of consecutive service failures that will cause a recycle of the server process. This is a catchall error handling routine that allows the dispatchers to terminate themselves if they receive multiple, consecutive, fatal error messages from service routines. Such errors should not occur consecutively, but if they do it indicates that the server process needs to be recycled or cleansed. A “Retry” message appears when this.

Scan Interval

Number of seconds between scans of the work queue when idle. The Scan Interval is necessary to detect messages published from two-tier connections since a message will be in the queue however the broker server does not receive a notice of the publication. So a Scan Interval is required to make sure that two-tier messages get processed in a timely manner. The Scan Interval is analogous to the Process Scheduler polling the Process Request table. In addition the Scan Interval detects messages that have been resubmitted after an error, for example.

Restart Period

Number of seconds between restart attempts on “started” items in work queue.

Handler Parameters

The following topics describe the parameters that apply to the handler process which include:

- PSBRKHND
- PSPUBHND
- PSSUBHND

Min Instances

Number of handler server processes started at boot.

Max Instances

Indicates 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 terminate itself and Tuxedo automatically restarts the process.

Recycle Count

Specifies the number of times the system executes each server before PeopleSoft intentionally terminates the process. Server processes must be intermittently recycled to clear buffer areas. The time required to recycle a server is negligible—occurring in milliseconds. Recycle Count

does not translate into a native Tuxedo parameter in the PSAPPSRV.UBB file. Instead the value is stored in memory and is managed by PeopleSoft.

Allowed Consec Service Failures

This option allows for dynamic server process restarts for service failures. To enable this option, enter a number greater than zero, and to disable this option enter 0. The default for this parameter is 2. The numerical value you enter is the number of consecutive service failures that will cause a recycle of the server process. This is a catchall error handling routine that allows the handler to terminate itself if it receives multiple, consecutive, fatal error messages from service routines. Such errors should not occur consecutively, but if they do it indicates that the server process needs to be recycled or cleansed. A “Retry” message will appear when this occurs.

Max Restarts

Indicates the maximum number of times that the server will attempt to restart a failed action.

Configuring PSMBSRV and PSMBHND

If you are using the Developer configuration template, you probably won’t need to perform too much tuning or configuring. Since it’s intended for development and demonstration purposes only, you can most likely accept all the defaults.

However, if you do need to make some changes, it’s important to know what to expect when you enter PSADMIN or your PSAPPSRV.CFG file. The Developers template blends the six typical messaging server processes into only two messaging server processes. The Developer template is not designed for production, and typically the domain would be running on a “normal” workstation or laptop, rather than a powerful server machine. Because of this, PeopleSoft blended the messaging servers for simplicity as well as to conserve system resources.

The Developer template offers the following two server processes to handle application messaging transactions:

- **PSMBSRV.** This server process is a combination of PSBRKSRV, PSSUBSRV, and PSPUBSRV. It absorbs all of the functionality of each of these server processes and performs each of the tasks as one “macro” messaging server. You can think of PSMBSRV as the dispatcher or listener (as in the Workstation Listener). PSMBSRV performs all of the listening and routing.
- **PSMBHND.** This server process is a combination of PSBRKHND, PSSUBHND, and PSPUBHND. Like PSMBSRV, it absorbs all of the functionality of each of these server processes and performs each of the tasks as one “macro” messaging server. You can think of PSMBHND as the handler. PSMBHND performs all of the actual publications and subscriptions.



Note. You can not dedicate the Developer messaging server to specific channels.

The following example shows the configuration section for the PSMBSRV. For parameter descriptions see those of PSBRKSRV.

Values for config section - PSMBSRV

Recycle Count=0

Allowed Consec Service Failures=0

Scan Interval=10

Restart Period=5

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

The following example shows the configuration section for the PSMBHND. For parameter descriptions see those of PSBRKHND.

Values for config section - PSMBHND

Min Instances=1

Max Instances=1

Service Timeout=0

Recycle Count=0

Allowed Consec Service Failures=0

Max Retries=5

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

CHAPTER 9

Administering the Application Messaging Gateway

The Application Messaging Gateway is a required component of your messaging system. It resides on the web server and fulfills the following functions:

- Maintains a directory of all nodes in the messaging system.
- Ensures that messages get routed to the proper destinations.
- Facilitates communication between nodes regarding delivery success and failure.
- Provides an integration point for publishing messages to a non-PeopleSoft node.

This discussion assumes that you've already installed the Application Messaging Gateway software to your web server as instructed in your Installation and Administration documentation.

In this chapter we address the procedures you must complete as you administer your Application Messaging Gateway.

Overview

The Web server is a vital link in the application messaging architecture. Just as the application server is required for any browser to connect to the database, for any node to publish a message to a PeopleSoft or non-PeopleSoft node, a Web server running the gateway must be involved. You need to configure at least one Application Messaging Gateway per system.



In some situations, configuring multiple gateways improves performance.

The Servlets

The Application Messaging Gateway involves the interaction of three separate PeopleSoft servlets. The three servlets are:

Gateway Servlet

This servlet contains a collection of handlers and is the component that actually performs the routing of messages to the target destinations. This is the servlet to which nodes perform an HTTP post of a message.

To test that the gateway servlet is running, you enter the following URL into your web browser:

```
<server_name>/servlet/gateway
```

A page reading "PeopleSoft Application Message Gateway" should appear. If it does not, then it indicates that the gateway or the web server is not running.

Configuration Servlet

This servlet is what you use to configure the gateway, which involves adding and loading handlers, modifying the Lookup Table, and so on. To access the configuration interface, enter the following URL into your web browser:

```
<server_name>/servlet/gateway.administration
```

Typically, only the messaging system administrators need to access the Configuration Servlet.

Reader Servlet

This servlet enables users to view the configuration information but not modify any configuration settings. To access the display-only interface, enter the following URL into your web browser:

```
<server_name>/servlet/gateway.handlers
```

Developers and power users may need to see a sub set of the information exposed in the configuration servlet. Rather than allowing them to view all of the connect information for a node and possibly disturb the configuration, you use the Reader Servlet to reveal information about the underlying messaging system. The Reader Servlet exposes the name of the node, the Machine address and port number, and the version of PeopleTools running on the node.

How it Works

The Application Messaging Gateway is a servlet designed to direct messages to their intended targets. The gateway servlet handles messages directed to both PeopleSoft and non-PeopleSoft nodes.

- Receives incoming messages from a PeopleSoft node and ensures that they arrive at the designated target PeopleSoft node.
- Receives incoming messages from a PeopleSoft node and sees that they are properly exposed to an external API thus enabling the PeopleSoft messaging system to include third party systems.
- Receives incoming messages from third party nodes and routes them to the designated target

PeopleSoft node.

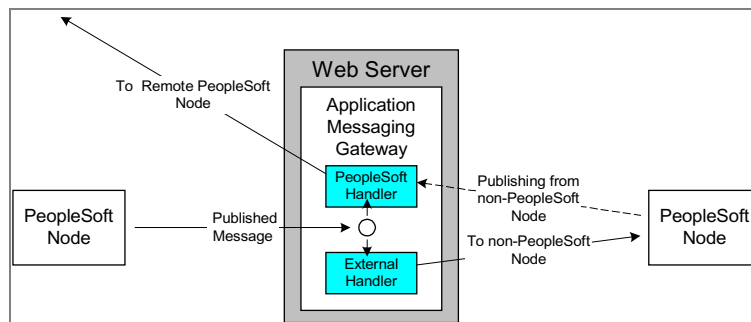
A PeopleSoft Node submits a published message to the Application Message Gateway in the form of an HTTP post. The routing rules that you've specified in Application Designer determine where the system routes the message. You can publish the same message to both a remote PeopleSoft node and an external system depending on your routing rules.



Note. PeopleSoft supports the ability to publish a message to PeopleSoft from a non-PeopleSoft node, and PeopleSoft also supports the ability to receive a message published by a third party node. To perform this requires some custom development work, which is beyond the scope of this administration discussion.

A handler performs the actual work of routing a message to the correct target node. PeopleSoft provides a collection of handlers for various tasks, but we expect that most sites will write custom handlers to suit their integration requirements.

For publishing messages to a PeopleSoft node, PeopleSoft provides the PeopleSoft Handler. Publishing to a PeopleSoft node includes publishing from one PeopleSoft node to another as well as publishing a message from a third party node to a PeopleSoft node.



Application Messaging Gateway Handlers

PeopleSoft also provides a handler that is designed for receiving a message published by a PeopleSoft node and writing it to a file for consumption by third party systems. This handler is called the Simple File Handler, and in the previous example the Simple File Handler would be the External Handler. Because you can write custom handlers to relay messages to a variety of third party systems and devices, the general term "External Handler" is meant to include any such handler.

Working with the Configuration Servlet

You use the configuration servlet to modify the settings for your gateway. To perform any of the tasks associated with configuring your gateway, you must first access the configuration servlet,

To launch the configuration servlet

1. Launch a supported web browser
2. Enter the URL of your configuration servlet.

For example,

```
<server_name>/servlet/gateway.administration
```

Adding Items to the Handler Directory

Each Handler that you intend to use in conjunction with the gateway needs to be registered, or loaded, using the PeopleSoft handler directory interface.

To add and load a handler

1. Launch the configuration servlet interface.
2. Click the **Add handler** button.

Handler	Status	Load	Unload	Configure	Delete
Add handler					

Add handler Button

3. On the Add Handler screen in the **Handler class:** edit box, enter the name of the handler class that is associated with the program you want to add/load.

You'll be prompted to supply the Java class that has implemented the proper PeopleSoft interfaces.

For example, to load the delivered Simple File Handler for writing messages to a file, enter the following:

```
psft.pt8.filehandler.SimpleFileHandler
```

Add Handler

Handler class:

Adding Handler Classes

4. Click **Save**.

Handler Directory					
Handler	Status	Load	Unload	Configure	Delete
psft.pt8.filehandler.SimpleFileHandler	Not loaded	Load			Delete

Add handler

Unloaded Handler

Notice that the handler now appears in the Handler Directory list, but the **Status** reads *Not loaded*. You can make that handler active by clicking the **Load** button. Once loaded, the gateway servlet loads the specified Java class in memory after searching the machine class path.

5. Click **Load**.

Handler Directory					
Handler	Status	Load	Unload	Configure	Delete
psft.pt8.filehandler.SimpleFileHandler	Loaded successfully		Unload	Configure	

Add handler

The Status column should read *Loaded successfully*. If you receive a “ClassNotFoundException error” as the status, the gateway servlet did not find the particular Java class you indicated. Make sure that the class path is correct. If necessary, you may need to reboot your web server.

During production, you may want to temporarily deactivate a handler. You deactivate a handler by clicking the **Unload** button. This removes the particular handler from the list of active registered handlers to which the gateway servlet can publish messages.

Deleting Handlers from the Handler Directory

When a handler is no longer required on the gateway, delete it using the following procedure.

To delete a handler

1. Locate the unneeded handler in the Peoplesoft handler directory list.
2. Click the **Delete** button on the same row as the handler information.

Configuring the PeopleSoft Handler

You use the PeopleSoft handler to publish messages to other PeopleSoft nodes. Each PeopleSoft node needs to be registered in the handler's Lookup Table so that the system can route messages

to the appropriate location. You administer the PeopleSoft handler and the lookup table using a browser.

Before you can publish nodes from one PeopleSoft node to another, you need to add, load, and configure the PeopleSoft Handler.

The following sections describe the tasks involved in configuring the PeopleSoft handler.

Adding Nodes to the Lookup Table

The functionality of the PeopleSoft Handler revolves around the lookup table. Every PeopleSoft node in the messaging system needs to be represented with an entry in the lookup table, and each time you add a new node to the system, you must also add a corresponding entry in the lookup table.

The lookup table is the table that contains the required connect information regarding every PeopleSoft node in a given messaging system. In this respect, the lookup table is the directory of a messaging system and all messages (except for those published locally and those intended for external systems) arrive at their intended destinations by way of the lookup table.

The information stored in the lookup table for a particular node consists of the following items:

- Node name.
- IP Address, or DNS name, and the JSL port. This appears in the following form

```
//<machine_name>:<JSL port>
```

- PeopleTools version running on the application server and database.
- PeopleSoft operator ID and password for logging on to the given database to which the application server is attached.

In order to add any of this information you must first launch the Manage Lookup Table interface.

To add a node to the lookup table

1. Launch the configuration servlet.
2. Click Add handler.
3. In the **Handler class** edit box, enter the following:

```
psft.pt8.psfthandler.PeopleSoftHandler
```

4. After adding the handler, click **Load**.
5. Click Configure.
6. Click Add a new node.

The **Add an address** interface appears.



Note. Click Cancel to return to the main page.

7. In the **Node** edit box, enter the name of the node you want to add.

Node
<input type="text" value="NODE_A"/>
e.g., EGEE_REMOTE

The name you enter must match the name defined in Application Designer for the Message Node definition.

8. In the **Machine address:port** edit box, add the appropriate information for the application server that's associated with the node's database.

For the machine address you can use the machine's DNS name or you can add the actual IP Address for the application server machine.

For the port number, enter the port number that specifies the Jolt Station Listener (JSL) on the application server. Remember that the Application Messaging Gateway sends messages to the application server through the Jolt communications layer.

Machine address:port#
<input type="text" value="//gsawyer061199:9000"/>
e.g., //bvoelke081099:9000

9. In the **Tools Version** edit box, enter the version of PeopleTools that the node is currently running.
10. In the **OPRID** and **Password** edit boxes, add the information required to connect to the node's application server and database.
 - Keep in mind that passwords and operator IDs are case sensitive.

After you have entered all of the appropriate values, click **Save address**.

Your interface should appear similar to the following example.

Manage Lookup Table

Node	Machine address:port#	Tools Version	OPRID	Actions		
NODE_A	//gsawyer061199:9000	8.0	PTDMO	Edit	Delete	Add

Add a new node

11. Repeat this process for every Node in your messaging system.

Multi-Application Server Nodes

In many cases you have more than one application server connecting to the database within a Node. For performance reasons and scalability, it is not uncommon to have two or three application servers associated with the same database.

Just like you need to add an entry for each Node in your messaging system, you also need to add an entry for each application server that belongs to a particular node.

The following procedure assumes that you've previously added a node entry in the lookup table.

To add multiple application servers to a node entry

1. Launch the Manage Lookup Table interface.
2. Locate the node entry to which you want to add another application server, and click **Add** in the Actions column.

Actions		
Edit	Delete	Add

The **Add an address** interface appears.

3. Enter the **Machine address:port**, **Tools Version**, **Operator ID**, and **Password**, as described in Adding Nodes to the Lookup Table.

The name of the Node automatically appears in the **Node** box.

Editing Nodes

If your environment changes and you need to modify entries in the Lookup table to reflect new values, such as machine addresses or passwords, you can do so by completing the following procedure.

To edit a node entry

1. Launch the **Manage Lookup Table** interface.
2. Locate the Node entry that you need to modify, and click **Edit** in the Actions column.

The **Edit an address** interface appears. Notice that the Node name appears automatically. You can not change the Node name once it is saved. If the Node name that appears in the Node box is not correct or current, you need to add a new entry and enter the current or correct name. You can, however, modify all of the other Node values.

3. Enter the appropriate changes in the **Machine address:port**, **Tools Version**, **OPRID**, or **Password** edit boxes.
4. After you have entered any modifications, click **Save change(s)**.

Deleting Nodes

If you no longer need a particular Node entry because it is incorrect or obsolete, you can easily remove it from the Lookup table.

To delete node entries from the lookup table

1. Launch the **Manage Lookup Table** interface.
2. Locate the Node entry that you want to delete, and click **Delete** in the **Actions** column.



The **Confirm deletion of entry** interface appears.

3. Make sure that the entry that appears in your browser is in fact the entry you want to delete, and click **Delete** to remove the entry from the Lookup table.

If you have second thoughts about deleting this entry, just press **Cancel**.

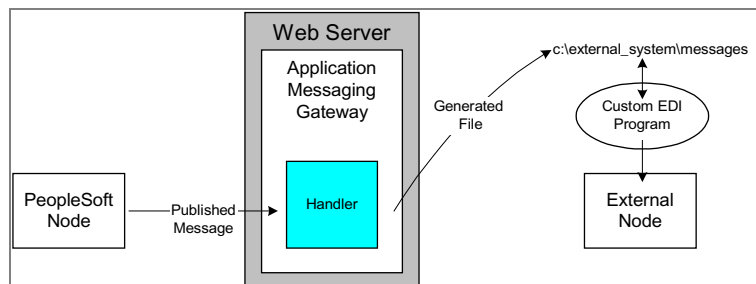
Configuring an External Handler

A handler is a Java program that “handles” or performs a particular file manipulation routine.

In order to push data out of your PeopleSoft messaging system you’ll need to set up a handler to complete this task. You also need to register the external nodes in the lookup table so that the servlet can deliver the message to the appropriate location.

As shown in the following example, you use the gateway to write a published message to a file in a specified location on your web server. A “handler” program actually processes the message and writes it to a file. You can think of a handler as a process that can consume a publication and expose it to third party systems. For instance the Handler that PeopleSoft ships, performs a routine that writes a PeopleSoft message to a file to be consumed by external nodes.

After the handler completes writing the message/file to the target directory, the message is “officially” out of the PeopleSoft system. To update the data in your external node, you must then develop a custom program, such as an EDI program, to consume the data stored in the generated file, and then ensure that the data tables in the external node reflects the current data.



Publishing a Message to a Third Party Node



PeopleSoft provides the functionality to write a published message to a file. Additional functionality, such as publishing to PeopleSoft from an external node, needs to be coded on an individual basis. To publish to PeopleSoft you need to construct an XML message in accordance with the PeopleSoft message syntax and then HTTP post the publication through our gateway.

The following sections describe the steps you need to complete in order to configure your gateway handlers.

Configuring a Handler

After you’ve added a handler to the handler directory, you need to configure it so that it writes the file to the appropriate location using the appropriate options.

To configure a handler

1. Launch the handler directory interface.
2. In the handler list, locate the handler that you need to configure, and click the **Configure** button.

The Simple File Handler Directory interface appears.

Simple File Handler Directory						
Node Name	Output File Name	Include Header?	Uncompress?	Base64 Decode?	Edit	Delete
<div>Add file handler</div>						
Back to Handler Directory						

- Click the **Add file handler** button.

This is where you add a list of nodes associated with a particular handler and where you set properties required for downloading a message to a file (for the SimpleFileHandler).

- Enter the **Node Name** and the **Output File Name** values.

Add File Handler				
Node Name	Output File Name	Include Header?	Base64 Decode?	Uncompress?
EXTERNAL_NODE_A	c:\PSFT_messages\file	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<div>Save</div> <div>Cancel</div>				

Adding a File Handler

The **Node Name** must match the name defined in Application Designer for the external node.

The **Output File Name** must reflect the exact location to which you want the file written and the correct file name that your custom program expects.



Note. The output location must be a directory on the web server.

- Choose the options you want.
 - **Include Header.** You have the option of excluding the Header portion of the XML message, which consists mainly of meta data tags and clear text. The message is (figuratively) split into two sections: a Header portion and a Data portion. The Data portion contains the “true” data section with data tags and panel buffer values. If you’re external program that reads the file does not require any Header information, then there’s no need to include it.
 - **Base64Decode and Uncompress.** Base64 encoding is required to allow a binary message to be sent between machines of different architectures. PeopleSoft uses Base64 to encode our compressed application messages so that the system is able to send messages to multiple platforms. We compress the messages for performance reasons. If you need the message decoded from Base64 and uncompressed, you have the option to indicate that. Choosing which option is appropriate depends on the program that consumes the file. If you have a preexisting program that expects to read an XML file from a particular directory on the web server, you would probably want to enable the Decompress & Base64 decode options. If, on the other

hand, you're writing your program from scratch, you might want to leave the file as-is (or just turn on Base64 decode), and have your custom program perform the decoding & decompression. This approach reduces the load on the web server.

CHAPTER 10

Application Message Monitor

You use the Application Message Monitor, or Message Monitor, to monitor the messaging queues in your system. It provides status information on individual messages and aids in error processing and resolution. The Application Message Monitor is designed for a system administrator, not the end user.

The Application Message Monitor is analogous to the Process Monitor in that it is a component used solely for the purpose of attaining the status of the underlying system.

With the Message Monitor, you can view message volume and status using pre-defined dimensions or views. You can also view messages by message content. Each view summarizes specific characteristics about your messages.

After reading this chapter you will be familiar with the Application Message Monitor's interface, how to use it, and how to interpret the error messages produced by errant messages. The best way to get to know the Message Monitor is to drill down into each view and notice how the information is organized while sorting through this information.

Before you can begin monitoring and resubmitting messages, it's a good idea to get familiar with all the options that the Application Message Monitor provides. This chapter introduces you to the Message Monitor interface and the tasks you can perform with it.

To access the Message Monitor, select **PeopleTools, Application Message Monitor**. There are three menu options associated with the Message Monitor: Use, Process, and Inquire. Select the Use menu to gain access to the main Message Monitor component. The Process menu provides additional message administration features, such as running a batch program to archive delivered messages and scheduling a workflow program that automatically notifies users of errors in their message channels. The Inquire menu contains the components that you transfer to from the Message Monitor for details.

Using the Message Monitor Component

To launch the main Message Monitor component select **PeopleTools, Application Message Monitor, Use, Application Message Monitor**.

The Application Message Monitor component appears. By default, the Overview page is active.

PeopleTools, Application Message Monitor, Use, Application Message Monitor

Before using the Message Monitor, you should set up security for it using permission lists. Make sure that the individuals who need access to it have the proper authority. To navigate to permission lists, select PeopleTools, Maintain Security, Use, Permission Lists.

You use permission lists to grant access to the Message Monitor pages and also to grant read and/or write access to specific message channels. Users can only view messages in channels to which they have read access, and any edit action requires write access.



For more information on granting access to the Message Monitor, see Message Monitor Permissions.

The main Message Monitor interface consists of the following pages:

Page	Description
Overview	Provides a high-level view of your system queues so that you can isolate particular areas and then drill down for further information.
Message Instance	Shows all messages that have been created for publication and/or subscription for any defined message nodes, both local and remote.
Pub Contracts	Shows outbound messages, or publication contracts, that are to be delivered to the remote message nodes with which the system is interacting.
Sub Contracts	Shows inbound messages, or subscription contracts, that have been delivered to the system from remote message nodes or the local message node itself.
Channel Status	The Channel Status page displays the status of the message channels defined in the system.
Node Status	Displays the status of the local message node. It also allows pinging remote nodes to determine their availability.
Queries	Provides a predefined set of queries (built with PS/Query) that enable you to view details about the message setup, message channels and message nodes in your system and generate reports.

The following sections describe what operations you can perform on each page and what type of information regarding your messaging system that each page provides.

Before You Get Started

Before you begin monitoring the numerous facets of your messaging system, there are a few general guidelines to follow to make sure that you quickly drill down to the information you need.

Filtering Messaging Information

Because the Message Monitor provides information from every aspect of your messaging system, you need to understand how to filter the information to reduce the number of items before you. For instance, rather than sifting through every message in the entire system, the Message Monitor enables you to sort by publish node, publish date/time, live and archived messages, and so on.

You filter the "result set" on the following pages in the Message Monitor:

- Overview
- Message Instances
- Pub Contracts
- Sub Contracts

Message Monitor has general filtering options that apply to each page where filtering applies. These options appear at the top portion of the pages. The following table provides a description of each general filtering option.

Criteria	Description
Publish Node	Every message is stamped with the node that publishes it. This control provides a drop-down list containing all the nodes defined in your system. The message monitor only allows you to view information for the local system (database). But the local database may have in its queues messages published not only by the local node, but also remote nodes. There is only one local node for a database.
Last	Enter a numerical value, and select Days, Hours, or Minutes from the associated drop-down list. This control enables you to narrow and broaden your monitoring scope based on intervals of time. By selecting None, you receive all messages currently in the system.
Archived	The Archived check box is a toggle switch that enables you to specify a search on your archived message data or you "live" message data. To search archived message data, select the checkbox. To search "live" message data, deselect the checkbox.

On the pages where filtering applies, you enter your filtering criteria in the Message Criteria section, and the output, or result set, from your criteria appears in the status grid directly below the filtering options.

Filtering options that apply to specific pages are discussed later. For example, the description for filtering options that are specific to viewing Overview information, appear within the discussion of the Overview page.

Maintaining Filter Criteria

Typically, you look for information on numerous occasions using the same filtering criteria. Often, it is just a matter of becoming familiar with the output and display of information when using a particular set of filtering criteria. Rather than having to reset filtering options each time you launch the Message Monitor, the system saves your filtering options so that the next time you use Message Monitor, your previous filtering choices are set automatically.

To save your Message Monitor filtering selections

1. Select the filtering options you desire on each page on which filtering applies (Overview, Message Instances, Pub Contracts, and Sub Contracts).
2. After you have selected the appropriate filtering options, 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, and the next time you log in and launch the Message Monitor the system displays the message data according to your most recent filtering selections.

Overview

The Overview page is the first page you see when you access the Message Monitor. You check this page to get a high-level overview of the status of your Message Instances, Publication Contracts, and Subscription Contracts.

Depending on the information you gather, you then drill down further into the Message Instances, Pub Contracts, and Sub Contracts pages.

Overview Message Instances Pub Contracts Sub Contracts Channel Status Node Status Queries

Message Criteria

Publish Node: QE_LOCAL Last: 1 Days ☐ Archived

*Queue Type: Message Instance *Group By: Channel Refresh

Channel Name	Error	New	Started	Working	Done	Retry	Timeout	Edited	Canceled
TREE_MAINT	0	0	0	0	1	0	0	0	0

View All First 1 of 1 Last

PeopleTools, Application Message Monitor, Use, Application Message Monitor, Overview

The following sections contain information concerning the filtering options that are specific to this page and how to interpret the information that this page displays.

Filtering Options

You can filter the information displayed on this page using the following criteria.

Criteria	Description
Queue Type	Specify the particular queue in which you are interested in monitoring. Your options are Message Instance, Publication Contract, or Subscription Contract.
Group By	Use this option to view by Channel or Message. After clicking refresh, notice that the label of the following section changes to reflect the option you selected from this drop-down list.

After you select your filtering options, click **Refresh** so that the display reflects your Message Criteria.

Status Columns

Regardless of what queue type you are monitoring, the columns in the Message Name/Channel Name list remain the same.

The following table presents the possible status messages that you may encounter using the Message Monitor. Being aware of what each status means aids in troubleshooting and system tuning. The following table provides a brief description of each status.

Status	Description
Error	An error occurred during processing. Manual intervention is required.
New	Either the item has been written to the database, but has not been dispatched yet, or the item has just been resubmitted.
Started	The dispatcher is in the process of passing the item to a handler, but the handler has not received it yet.
Working	The handler has accepted the item and is currently processing it.

Done	<p>The handler successfully processed the item. Depending on the type of process you are monitoring, this status indicates different outcomes.</p> <p>Message Instance. All contracts have been created.</p> <p>Publication Contracts. The publication has been successfully received by the subscribing node</p> <p>Subscription Contracts. The subscription process ran successfully.</p>
Retry	The system encountered an intermittent error during processing. The system retries messages with this status automatically.
Timeout	Before a message transaction could be completed successfully, the system timed out.
Edited	The publication data for the item has been edited. Processing will not resume until you resubmit the item.
Cancelled	The item has been cancelled. The system can't process the item until you resubmit.

The numeric values in the status columns indicate the number of messages associated with the status for that column and with the channel name or message name for that row. If a value greater than 0 appears in any of the status columns for a channel Name or a message name, you click on the value, which is a hypertext link.

The hyperlink enables you to quickly gather more information by taking you to the appropriate page in the Message Monitor and populating that page with the messages corresponding to the number you clicked. For instance, if you are monitoring the Queue Type of Message Instance and you click on a non-zero value (say 5) in the Error status column, the system automatically opens the Message Instances page and populates it with the 5 messages with Error status.

Message Instances

The Message Instances page enables you to monitor the status and details related to the individual message instances that exist in either your live or archived system. Every publication and subscription contract is associated with a message instance.

Publishing Node	Channel	Pub ID	Message	Status	Time Stamp
QE_LOCAL	TREE_MAINT	2	TREE_CHANGE	Done	07/07/2000 9:33:20AM

PeopleTools, Application Message Monitor, Use, Application Message Monitor, Message Instances

Filtering Options

You can filter the information displayed on this page using the following criteria.

Criteria	Description
Channel Name	To view message instances within a specific channel, select the appropriate channel value from the Channel Name dropdown list.
Message Name	To view the instances of a particular message definition, select the appropriate message value from the Message Name dropdown list.
Status	To view message instances by status, select the status criteria from the Status dropdown list. The status options reflect the status columns that appear on the Overview page.
Order By	The Message Instances page enables you to monitor the status and details related to the individual message instances that exist in either your live or archived system. Every publication and subscription contract is associated with a message instance.
Asc/Desc	For the Order By selection, these radio buttons allow you to specify an ascending or Descending order. For an ascending order, select Asc, and for a descending order, select Desc. If the Order By value is numerical, as in Pub ID or time Stamp, then the order will ascend or descend in numerical order. On the other hand, the system sorts character values alphabetically.

After you select your filtering options, click **Refresh** so that the display reflects your Message Criteria.

Output

The output in the status grid of the Message Instance page enables you to view the status of message instances by Channel, Date, Status, and so on.

After you locate a particular message instance, you click the **Details** link if you need to view more detailed information associated with the message instance.

The Details button automatically launches the Message Details component or a custom, application-specific page where you can view and correct errors.

Whether you use the Message Details component or a custom page to edit a message, depends on what properties the developer selected for the Message Definition in Application Designer. The Details button invokes a Transfer Page to the component specified in the Message Definition, Message Properties, Error Viewing/Correction option.



For more information on message properties, see Viewing and Changing Message Properties.

Pub Contracts

The Publication Contracts page displays all messages that require delivery to a remote node. The system does not create Publication Contracts for routing to the local node.

PeopleTools, Application Message Monitor, Use, Application Message Monitor, Pub Contracts

Filtering Options

You can filter the information displayed on this page using the following criteria.

Criteria	Description
Channel Name	To view message instances within a specific channel, select the appropriate channel value from the Channel Name dropdown list.
Message Name	To view the instances of a particular message definition, select the appropriate message value from the Message Name dropdown list.
Status	To view message instances by status, select the status criteria from the Status dropdown list. The status options reflect the status columns that appear on the Overview page.
Order By	The options available in the Order By dropdown list correspond to the columns in the status grid below the Message Criteria section. You can select Publishing Node, Channel, Pub ID, Subscriber Node, Message, Status, and Time Stamp.
Asc/Desc	For the Order By selection, these radio buttons allow you to specify an ascending or descending order. For an ascending order, select Asc , and for a descending order, select Desc . If the Order By value is numerical, as in Pub ID or Time Stamp, then the order will ascend or descend in numerical order. On the other hand, the system sorts character values alphabetically.

After you select your filtering options, click **Refresh** so that the display reflects your Message Criteria.

Output

The output in the status grid of the Pub Contract page enables you to view the status of individual publication contracts by Channel, Date, Status, and so on.

After you locate a particular publication contract, you click the **Details** link if you need to view more detailed information associated with the contract.

The Details button automatically launches the Message Details component or a custom, application-specific page where you can view and correct errors.

Whether you use the Message Details component or a custom page to edit a message, depends on what properties the developer selected for the Message Definition in Application Designer. The Details button invokes a Transfer Page to the component specified in the Message Definition, Message Properties, Error Viewing/Correction option.

Sub Contracts

The Sub Contracts page displays all messages subscribed to by the local node. The display does not reveal subscription contracts for remote nodes.

PeopleTools, Application Message Monitor, Use, Application Message Monitor, Sub Contracts

Filtering Options

You can filter the information displayed on this page using the following criteria.

Criteria	Description
Channel Name	To view message instances within a specific channel, select the appropriate channel value from the Channel Name dropdown list.
Message Name	To view the instances of a particular message definition, select the appropriate message value from the Message Name dropdown list.
Status	To view message instances by status, select the status criteria from the Status dropdown list. The status options reflect the status columns that appear on the Overview page.

Order By	The options available in the Order By dropdown list correspond to the columns in the status grid below the Message Criteria section. You can select Publishing Node, Channel, Pub ID, Message, Subscription, Status, and Time Stamp.
Asc/Desc	For the Order By selection, these radio buttons allow you to specify an ascending or descending order. For an ascending order, select Asc, and for a descending order, select Desc. If the Order By value is numerical, as in Pub ID or Time Stamp, then the order will ascend or descend in numerical order. On the other hand, the system sorts character values alphabetically.

After you select your filtering options, click **Refresh** so that the display reflects your Message Criteria.

Output

The output in the status grid of the Sub Contract page enables you to view the status of individual subscription contracts by Channel, Date, Status, and so on.

After you locate a particular subscription contract, you click the **Details** link if you need to view more detailed information associated with the contract.

The Details button automatically launches the Message Details component or a custom, application-specific page where you can view and correct errors.

Whether you use the Message Details component or a custom page to edit a message, depends on what properties the developer selected for the Message Definition in Application Designer. The Details button invokes a Transfer Page to the component specified in the Message Definition, Message Properties, Error Viewing/Correction option.

Channel Status

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, first you should check the Channel Status page to make sure the channel is running.

There are occasions where you need to shut down a particular node to perform a specific maintenance task. Typically, a node contains numerous channels. The ability to shut down a particular channel as needed means that the other channels on the node continue processing messages undisturbed.

To determine the status of a particular channel, locate it within the Channel Name list, and then check the corresponding value in the Status column.

Overview Message Instances Pub Contracts Sub Contracts Channel Status Node Status Queries		
Channel Name	Status	
BUS_EXP_MSG_CHNL	Running	Pause
EMAIL_CHNL	Running	Pause
JOB_CHANNEL	Running	Pause
MARKET_RATES	Running	Pause
MARKET_RATE_LOAD	Running	Pause
QE_GAME_CHNL	Running	Pause
QE_PLYRTRD_CHNL	Running	Pause
QE_PLYR_CHNL	Running	Pause
QE_UPS_TM_CHNL	Running	Pause
TREE_MAINT	Paused	Run
View All First 1-10 of 11 Last		
Previous tab Next tab		
Overview Message Instances Pub Contracts Sub Contracts Channel Status Node Status Queries		

PeopleTools, Application Message Monitor, Use, Application Message Monitor, Channel Status

Notice that the button to the right of the Status value reflects the opposite value of the Status value. To reverse the status of the channel, as in to pause a running channel or to start a paused channel, simply click the button that corresponds to the channel name.

Node Status

The Node Status page enables you to perform the following:

- View the pause times for the local node.
- Add a pause time for the local node.
- Delete an existing pause time for the local node.
- Test the local node.
- Ping a remote node.

PeopleTools, Application Message Monitor, Use, Application Message Monitor, Node Status

Scheduled System Pause Times For Local Node

A "Pause Time" refers to an interval of time in which the message node becomes inactive. When the pause time begins, the messaging node is effectively shut down until the pause time is scheduled to end.

There are times when you need to schedule a pause time so that you can perform regular 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 machine as a particular message node every Monday morning from 12:05 AM to 3:30 AM. 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 12:00 AM to 4:00 AM. A pause time like this enables you to make sure batch run has ample system resources to complete successfully within the desired batch window.

During a pause time, messages won't be published or received by the local system. When your system is 'Paused', the node can't accept the message sent to it, and the publishing node must attempt to send the message again later. The publishing node continues to send the message until it exceeds the local 'Time Out Period.' When this happens, the message assumes a 'Timeout' status in the publisher's message queue. Keep in mind that 'Time Out Period' is an attribute of the publication channel, not the subscription channel.

If your system attempts to send a message while the message node is paused, the system writes the message to the Pub/Sub queues, but the system can't physically publish the message until the system is no longer in the paused state.



Pause times do not appear in an Application Designer upgrade project; you cannot upgrade them.

To add a node pause time

1. Click Add Pause.

The Set Message Node Pause Interval page appears.

Node Status, Add Pause

2. Select a day of the week from the **Start Day** dropdown list.

This value must reflect the day you want the pause to start.

3. Enter a value in the **Start Time** edit box.

The value you enter here applies to your Start Day. Enter the time of day that you want the pause time to begin.

4. Select a day of the week from the **End Day** dropdown list.

This value must reflect the day you want the pause to start.

5. Enter a value in the **End Time** edit box.

The value you enter here applies to your End Day. Enter the time of day that you want the pause time to end and normal processing to resume.

6. After you have entered the appropriate start and end values to define your pause interval, click **OK**.

To delete an existing pause time

1. In the pause time list, locate the pause time (interval) that you want to delete.
2. Click the **Delete** button to the right of the entry in the pause time list.

To test the local message node

1. Make sure you are logged onto the node that you want to test.
2. Click the **Test Node** button.

Depending on the status of the node, you receive one of the following messages:

Node is paused.

Or

Node is not paused.

Testing the Availability of a Remote Node

To check to see if a remote node is currently running and able to receive messages from the local node, you ping the remote node from the Node Status page. A successful ping indicates that the remote node is currently available. An unsuccessful ping could indicate that the node and/or the gateway are not running.

To ping a remote node

1. In the **Ping a Node to Determine Availability** section, select the node from the **Message Node Name** dropdown list.

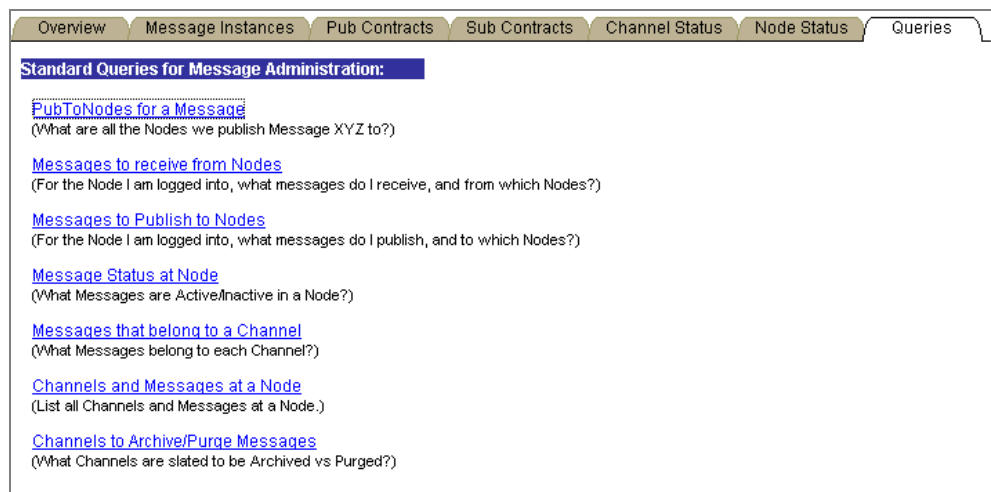
The **Location** column in the grid below reveals the locations defined for the node.

2. Click the **Ping Node** button.
3. The **Result** column displays the ping responses for the corresponding locations.

Queries

When you have numerous messages, nodes, and channels in your messaging system keeping track of all of the definitions and routings can become an unwieldy task. The Queries page is designed so that the system administrator is able to gather information about the messaging system meta-data without having to access the message definitions, channel definitions, or node definitions in Application Designer.

The queries take advantage of the URL interface for PeopleSoft Query (PSQuery) and Process Scheduler's Report Manager.



PeopleTools, Application Message Monitor, Use, Application Message Monitor, Queries

Because PSQuery is involved, anyone running the queries requires the appropriate security permissions to do so.



For more information on PSQuery see Using PeopleSoft Query, and for more information on Process Scheduler see Process Scheduler Basics.

These queries are not intended to be run every day or even on a regular basis. You might find that they are most useful for providing auditing information shortly after an implementation or upgrade. PSQuery enables you to view the information with Excel, which makes sharing the information with colleagues easier.

The following table contains descriptions of each of the available queries.

Query	Description
PubToNodes for a Message	This query reveals all of the nodes to which the system publishes a particular message. For instance, the query would show all of the nodes that receive messages regarding changes in the Personal Data tables?
Messages to receive from Nodes	This query shows all of the messages that the local node receives (subscribes to), and it also shows what nodes publish each particular message. For example, if you were curious as to what messages the Chicago HRMS database gets and what nodes are sending them, you run this query.
Messages to Publish to Nodes	This query shows what messages the local node publishes and to which nodes. For example, if you were curious as to what messages the Chicago HRMS database publishes and what nodes are receiving them, you run this query.
Message Status at Node	This query shows what messages are active or inactive in the local node.
Messages that belong to a Channel	This query shows what messages belong to each Channel definition. For example, if you were curious as to what messages are grouped together for ordering and routing, run this query.
Channels and Messages at a Node	This query enables you to get an overview of the messaging topology in a particular node/database. The output reveals the message status for a node as well as showing which messages are associated with which channels.
Channels to Archive/Purge Messages	This query shows which channels should have their messages archived and which channels should have their messages purged. You can customize the query to prompt for criteria.

Working with Message Monitor Processes

The following sections describe the batch processes that PeopleSoft provides to assist in notifying you of errors and archiving messages in your messaging system. You access the interface for invoking each of these programs by selecting **PeopleTools, Application Message Monitor, Process**.

Under the Process menu are two options:

- Error Notification
- Message Archiving

Error Notification

Although you can easily use the Message Monitor to scan your system for messages, that approach requires you to launch Message Monitor on a scheduled basis to search for any issues affecting your messaging system. Rather than using this labor-intensive approach, PeopleSoft provides an Application Engine program, named PT_AMM_WF, that you schedule to run on a recurring basis.

To enable the workflow notification functionality, you need to have the following items in place within your security definitions:

- Grant access to the PT_AMM_DUMMY component interface using PeopleTools, Maintain Security, Use, Permission Lists, Component Interface.
- Assign Users to the APP_MSG_ADMINISTRATOR role using PeopleTools, Maintain Security, Use, Roles. PeopleSoft delivers this role "out of the box."
- Users with assigned to the APP_MSG_ADMINISTRATOR role need to have their e-mail address added to their user profile so that they can receive the notification. You complete this task using PeopleTools, Maintain Security, Use, User Profiles.

You invoke and/or schedule the program to run by using the Error Scan PubSub page.

PeopleTools, Application Message Monitor, Process, Error Notification

The following table provides a step-by-step process revealing what the Application Engine program, PT_AMM_WF, scans for, how it notifies administrators, and what administrators should do after receiving an error notification.

Step	Task	Description
1	Query Message Queues	The Application Engine program, PT_AMM_WF, 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 triggers a Workflow notification, which the system sends, by default, to all users that qualify for the query Role APP_MSG_ADMINISTRATOR at runtime. PeopleSoft supplies this Role as part of the delivered system data. As delivered, 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 get notified.
3	Resolve Issue	After an administrator receives the workflow notification by email, they check their worklist to find a new worklist item reflecting the problematic message. To access the message, the administrator clicks the item in the worklist. The link leads to the Message Details component. The component is presented with the errored message loaded.

To run PT_AMM_WF

1. Select PeopleTools, Application Message Monitor, Process, Error Notification.
2. Select an existing Run Control ID, or add a new one using the **Add** button.

The Error Scan PubSub page appears.

3. Select a Process Frequency.

You have the following choices:

- **Process Once.** If you only want run PT_AMM_WF once, manually, on and as needed basis, select this option.
 - **Process Always.** If you want PT_AMM_WF to always run, constantly, select this option.
 - **Don't Run.** If you want to disable a recurring PT_AMM_WF run, select this option.
4. Add a Request ID and Description.

This is where you add attributes to uniquely identify a Run Control. You only see it when you have a list of Run Controls.

5. Click **Run**.
6. Click **OK** on the Process Scheduler Request page to submit the process.

Message Archiving

For performance and general maintenance reasons you want to archive older messages to clear space on your live messaging tables. PeopleSoft provides an Application Engine program, APPMSGARCH, which scans all of the messaging tables in the system and removes all messages with a status of "DONE". You use the Run Archive page to invoke the archive process.

PeopleTools, Application Message Monitor, Process, Message Archiving



Using APPMSGARCH to archive your message data is the batch approach. You can also archive individual messages online using the Message Monitor.

To run APPMSGARCH

1. Select PeopleTools, Application Message Monitor, Process, Message Archiving.
2. Select an existing Run Control ID or add a new one.
The Run Archive page appears.
3. Make sure the appropriate Run Control ID appears on the page, and click **Run**.
4. On the Process Scheduler Request page make the appropriate selections and click **OK**.

Using the Message Details Component

The message details component allows you to gather in depth information about a specific message.

The message details component contains the following pages:

- message properties
- message errors
- XML message viewer
- structured message viewer

You can navigate to the message Details component using the PeopleTools menu structure by selecting **PeopleTools, Application Messaging, Inquire, Message Details**. However, in most cases you access the message details component by clicking the Details button on one of the pages in the Message Monitor component or by clicking on a worklist entry representing an error notification.

You use the message details component to view detailed information, such as field data, contained in specific messages. It also enables you to perform tasks such as, correct errors, and resubmit messages.

The following sections describe each page of the message details component.

Message Properties

The Message Properties tab displays information pertaining to the Message Instance and all associated Publication and Subscription Contracts.

Message Properties | Message Errors | XML Message Viewer | Structured Message Viewer

Message Instance Information | Actions | Refresh

Pub Node: QE_LOCAL | Pub Process: Tree Manager | Resubmit

Channel: TREE_MAINT | Time Stamp: 07/07/2000 9:33:20AM | Cancel

Pub ID: 2 | Publisher: PT8 | Delete

Message: TREE_CHANGE | Status: Done

Publication Contracts | View All | First | 1 of 1 | Last

Actions | Information

Subscriber Node | Status

Subscription Contracts | View All | First | 1 of 1 | Last

Actions | Information | Tuxedo Client IDs

Subscription | Status

PeopleTools, Application Message Monitor, Inquire, Message Properties

The following topics briefly describe the type of information that appears in each grid.

Message Instance Information

The Messaging Instance Information grid provides general information pertaining to a particular message to assist in troubleshooting.

Message Attribute	Description
Pub Node	The node that published the message.
Channel	The channel to which the message is associated.
Pub ID	The publication ID.
Message	The name of the message.
Pub Process	The process that published the message; the name of the component that published the message.
Time Stamp	The time that the message instance was last processed.
Publisher	The publisher of the message. This is usually the UserID of the person in the publishing system who triggered the message publication.
Status	Status of the message such as Done, Error, Started, and so on.

The Actions group contains controls that apply to the Message Instance and all associated Publication and Subscription contracts. The following table describes when each button applies.

Action	Description
Resubmit	Applies to messages with a status of Time Out, Error, Edited, and 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 <i>Edited</i> . When you resubmit the message, the status changes, yet again, to <i>New</i> .
Cancel	Applies to messages with a status of New, Retry, Time Out, Error, and Edited.
Delete/Archive	Applies to messages with a status of Done or Cancelled.

Publication Contracts

The following sections describe the information provided in the Publication Contracts grid.

Actions

This 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.

Information

The Information tab provides the following additional information.

Item	Description
Time Stamp	Indicates 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 on the local application server. It shows the PID of the PSMSGHND (handler) that created the contract.

Subscription Contracts

The following sections describe the information provided in the Subscription Contracts grid.

Actions

This tab reveals the status of a particular subscription contract.

Information

Item	Description
Time Stamp	Indicates 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 on the server.
SubProcID	Shows the subscription process ID associated with the subscription contract for identification purposes.

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.

PeopleTools, Application Message Monitor, Inquire, Message Errors

You can click on the View button on any error row to automatically navigate directly to the problematic data field. The error information appears in the tree format. This enables you to locate errors and resolve them quickly.

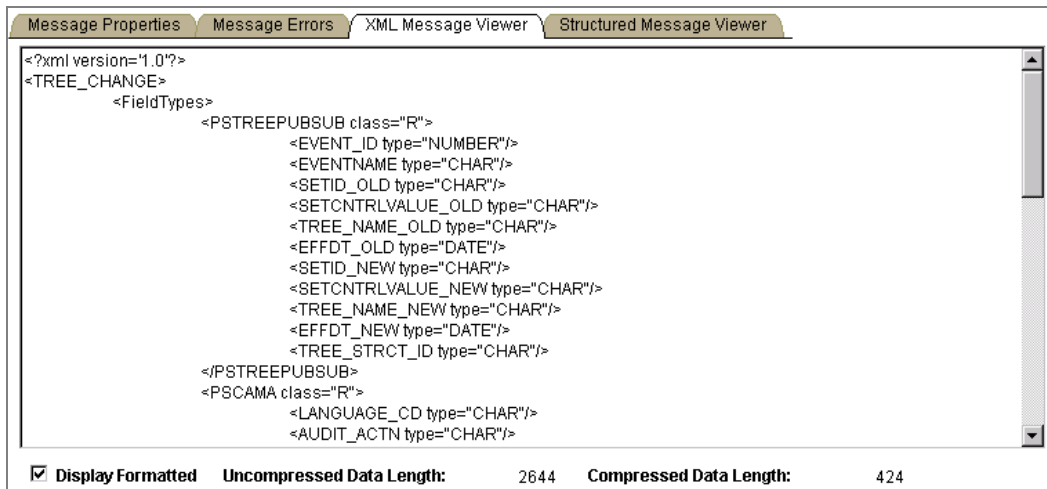


For both subscription contracts and publication contracts, error messages for all contracts appear in the grids, regardless of which individual contract you originally selected.

XML Message Viewer

The XML Message Viewer page enables you to view the raw XML code. This tab is where 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 is easier to read and to locate specific values or tags.



PeopleTools, Application Message Monitor, Inquire, XML Message Viewer



You cannot edit the raw XML and save it.

Structured Message Viewer

The Structured Message Viewer contains a tree control that dynamically displays the message structure and data. The Structured Message Viewer shows the structure of the message similar to how it appears within Application Designer.

Using the Structured Message Viewer

Click the plus and minus signs to expand/collapse individual nodes of the tree. Use the **Collapse All** button to close the entire tree.

Field	Value
EVENT_ID	5
EVENTNAME	TreeChangeMsg
SETID_OLD	
SETCNTRLVALUE_OLD	
TREE_NAME_OLD	PORTAL
EFFDT_OLD	2001-01-17
SETID_NEW	
SETCNTRLVALUE_NEW	
TREE_NAME_NEW	PORTAL
EFFDT_NEW	2001-01-17
TREE_STRCT_ID	PORTAL

PeopleTools, Application Message Monitor, Inquire, Structured Message Viewer

To edit a data item

1. Locate the message data you need to modify.
2. Make the appropriate changes in the edit box.
3. Click **Save** to commit your changes to the Message Queue.
4. Resubmit the message using the Message Properties page.

To print a structured message

1. Click **Print**.

The system invokes a new instance of your browser.

2. In the new browser instance, select **File, Print**.

Customizing the Structured Message Viewer

As delivered, the Structured Message Viewer displays messages up to four levels deep. The maximum levels allowed by PeopleSoft components is three. So for messages published from component data, the number of levels in the Structured Message Viewer is adequate.

It is possible to publish messages that are more than four levels deep. To view all levels of messages that exceed four levels, you can customize the Structured Message Viewer as follows using the Application Designer. In the following instructions, the number suffixes assume that you are extending the Structured Message Viewer one level—from four levels to five levels.



A message with four levels includes levels zero through four (0-4).

The following procedure assumes a working knowledge of Application Designer.

To customize the structured message viewer

1. Create the following field definitions:

- **ROWNAME9.** Open ROWNAME7 and Save As ROWNAME9.
- **AMM_ICON9.** Open AMM_ICON8 and Save As AMM_ICON9.
- **AMM_PB7.** Open AMM_PB6 and Save As AMM_PB7.
- **AMM_PB7M.** Open AMM_PB6M and Save As AMM_PB7M.
- **AMM_TEXT6.** Open AMM_TEXT5 and Save As AMM_TEXT6.
- **APMSG_FETCH6.** Open APMSG_FETCH5 and Save As APMSG_FETCH6.

2. Add the fields you created in Step 1 to the record: AMM_TREE_WS.

3. Add record fields to page.

Open page AMM_TREEVIEW. Turn on Object Inspector by selecting View, Show Object Inspector. Add the fields created in Step 1 to the Scroll Area on the page. The size, location, and order of the panel fields should conform to the previous levels. The Object Inspector displays the size and location of the selected panel field. The Order tab shows the order of the panel fields. Each level is staggered horizontally by 22 pixels.

4. Add and modify PeopleCode.

- Copy PeopleCode from AMM_TREE_WS.AMM_PB6.FieldChange and paste to AMM_TREE_WS.AMM_PB7.FieldChange.
- Replace PeopleCode in AMM_TREE_WS.AMM_PB6.FieldChange with a copy from AMM_TREE_WS.AMM_PB5.FieldChange.
- Copy PeopleCode from AMM_TREE_WS.AMM_PB6M.FieldChange and paste to AMM_TREE_WS.AMM_PB7M.FieldChange.
- Replace PeopleCode in AMM_TREE_WS.AMM_PB6M.FieldChange with a copy from AMM_TREE_WS.AMM_PB5M.FieldChange.
- Copy PeopleCode from AMM_TREE_WS.AMM_TEXT5.FieldChange and paste to AMM_TREE_WS.AMM_TEXT6.FieldChange.
- Copy PeopleCode from AMM_TREE_WS.APMSG_FETCH5.FieldChange and paste to AMM_TREE_WS.APMSG_FETCH6.FieldChange.

- In AMM_DERIVED.ARCHIVE_BTN.FieldFormula PeopleCode, locate the following statement:

```
/* Set maximum number of levels allowed by the Structured Message Viewer in here.
```

```
Levels are numbered starting at 0. So &MaxLevelNo = 4 corresponds to 5 levels. */
```

```
&MaxLevelNo = 4;
```

Set &MaxLevelNo to the appropriate number (the number of levels that need to be viewed).

Purging the Messaging Tables

PeopleSoft 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 what each script is designed to accomplish.

<i>Script Name</i>	<i>Description</i>
AppMsgPurgeAll.dms	This script deletes the queue data from every archive or live message table in the database. Typically, you run this script after an upgrade or while switching from your demonstration to your production environment.
AppMsgPurgeArchive.dms	This script deletes the queue data from every archive message table in the database.
AppMsgPurgeLive.dms	This script deletes the queue data from every live message table in the database.

Printing a Message

On occasion you may want to print a message for closer examination or for sharing with colleagues.

To print a message

1. Select PeopleTools, Application Message Monitor, Inquire, Print Message.
2. On the search page, select the message you want to print.

The message appears on a page in a format similar to that of the Structured Message and formatted XML display.

3. Select **File, Print** from your browser menu bar.

Using the Message Monitor Component Interface

Not all of the Message Monitor functionality is exposed, but PeopleSoft does provide a collection of inquiry methods.

The name of the component interface you use to extract information from the Message Monitor is MSGSTATUSSUMMARY. 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.

The following list contains the user-defined methods you use to extract information:

- FillPubConByMsg():
- FillPubConByChannel()
- FillSubConByMsg()
- FillSubConByChannel()



For more information on our integration technologies, such as component interfaces, see [\[ADD LINK TO INTEGRATION DOC\]](#).

Following example shows some sample of ASP code that accesses the MSGSTATUSSUMMARY component interface with COM.

```
'Create a peoplesoft session

Set oSession = server.CreateObject ("PeopleSoft.Session")

nStatus = oSession.Connect(1, oConnectString, oUserName, oPassword,0)

'Get the skeleton of the APPMSGMON CI

Set oCI = oSession.GetCompIntfc("MSGSTATUSSUMMARY")

'get an instance of the CI

nStatus = oCI.Get()

'execute the method to fill the collection

If oChoice = 1 then

    nStatus = oCI.FillPubConByChannel()

    '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
```

Index

%

%MaxMessageSize() 4-5

A

- adding message nodes 2-1
- administration
 - Application Messaging Gateway 9-1
 - messaging server 8-1
- alias field in message definitions 2-12
- Application Designer
 - messaging objects 2-1
- application message monitor 10-1
- Application message monitor
 - interface 10-1, 10-2
- application messaging
 - advantages 7-2
 - application server 7-7
 - architecture 7-1, 7-4
 - creating a test message 2-18
 - creating messages 1-1
 - determining message status 10-5
 - local nodes 7-6
 - message channel 7-4
 - message node 7-4
 - message server 7-7
 - monitoring 10-1
 - nodes 7-5
 - pausing system 10-12
 - publish-and-subscribe model 1-1
 - publishing to a local node 7-11
 - publishing to a remote node 7-12
 - publishing to a third party 7-14
 - publishing to remote nodes 9-1
 - remote nodes 7-6
 - required components 7-4
 - sample scenarios 7-11
 - terminology 7-2
- Application Messaging Gateway 6-1, 7-10
 - adding nodes 9-5
 - administration 9-1
 - external gateway 9-9
 - lookup table 9-6
 - overview 9-1
 - PeopleSoft Gateway Servlet 7-3
- application server
 - configuring the message server 8-1
 - message server 7-7

- brokers and contractors 7-8

- dispatchers and handlers 7-9

- messaging server processes 7-7
- removing messaging servers 8-15

- architecture
 - application messaging 7-1
- archiving messages 10-18
- audit action codes 3-5, 6-27
 - checking the value 3-6
- authenticity of messages 2-3

C

- CD-ROM
 - ordering ix
- changing message properties 2-17
- channel lists
 - editing 8-12
- channels
 - monitoring 10-10
- component interface for message monitor 10-27
- compressed messages 6-17
- configuration servlet 9-2
- configuration template
 - developer 8-19
- cross node messaging 2-19

D

- dedicated message servers 8-2
 - adding 8-2, 8-3
 - assigning 8-3
- dedicated messaging servers
 - channel lists 8-12
 - configuring 8-12
 - PSADMIN 8-17
 - removing 8-15
- defining message node 2-1
- dispatcher parameters 8-17

E

- EditError property 5-2
- error notification in message monitor 10-16
- error tracking in application messaging 5-1
- errors
 - correcting 5-3
 - correcting in a page 5-6

- correcting in message viewer 5-6
- ExecuteEdits 5-1
- Exit function 5-2
 - Exit(1) 5-3
- external gateway 9-9
- external gateway servlet 9-9

F

- filtering
 - message monitor 10-7
- filtering in message monitor 10-3

G

- gateway
 - handler directory 9-3
- gateway servlet 9-2

H

- handler
 - configuring 9-10
- handler directory 9-3
 - adding items 9-4
 - configuring a handler 9-10
 - deleting items 9-5
- handler parameters 8-18

I

- inserting a message subscription 2-14
- inserting records in message version 2-13

L

- language code 6-26
- local node
 - scheduling system pause time 10-12
- lookup table 9-6
 - adding multi-application server nodes 9-8
 - adding nodes 9-6
 - editing nodes 9-8
 - removing nodes 9-9

M

- maximum message size
 - checking for 4-5
- message
 - authenticating 6-10
 - defined 7-3
- message channel 1-1, 7-4

- adding messages to 2-9
- defining 2-3
 - how it's used at runtime 2-7
- partitioning 2-9
- setting properties 2-7
- message channels
 - performance 8-2
- message definition 1-1
 - list view 2-11
 - structure view 2-11
- message designer
 - views of 2-11
- message details component 10-19
- message errors page in message monitor 10-22
- message instances
 - monitoring 10-6
- message monitor
 - component interface 10-27
 - error notification 10-16
 - errors page 10-22
 - filtering information 10-3
 - filtering options 10-7
 - maintaining filtering criteria 10-4
 - message archiving 10-18
 - message details component 10-19
 - overview page 10-4
 - printing a message 10-26
 - processes 10-16
 - purging message tables 10-26
 - queries page 10-14
 - structured message viewer 10-23
 - XML message viewer 10-23
- message node 1-1, 7-4
 - defined 7-4
 - defining 2-1
- message node designer 2-1
- message nodes
 - monitoring status 10-11
- message objects
 - order defined 2-1
- message properties
 - viewing or changing 2-17
- message Queue
 - defined 7-3
- message status 10-5
- message subscription
 - inserting into message definition 2-14
- message versions 2-12
 - default structure 2-12
 - inserting record in 2-13
- message viewer
 - launching 10-19
 - printing messages 10-24
 - using 10-19
- messages
 - compressed 6-17
 - defining or creating 2-11
 - monitoring 10-1

- publishing 3-1
- subscriber testing 4-7
- subscribing to 4-1
- messaging errors 5-1
- messaging server
 - administration 8-1
 - administration menu 8-7
 - channel lists 8-12
 - dedicated 8-3
 - default 8-3
 - demonstration 8-19
 - removing 8-15
- messsage server 7-7
 - broker 7-8
 - contractor 7-8
 - dispatchers 7-9
 - handlers 7-9
 - server processes 7-7

N

- node access password 2-3
- nodes
 - local 7-6
 - lookup table 9-9
 - remote 7-6
 - remote PeopleSoft node 7-6
- Nodes 7-5
 - non-PeopleSoft nodes 7-6

O

- OnRoutePublish PeopleCode 2-5
 - sample of 2-5
- OnRouteSubscribe PeopleCode 2-6
 - sample of 2-6

P

- parameters for dispatcher process 8-17
- parameters for handlers 8-18
- partitioning a message channel 2-9
- PeopleBooks
 - CD-ROM, ordering ix
 - printed, ordering ix
- PeopleCode
 - example of single node publish and subscribe 2-16
 - OnRoutePublish 2-5
 - OnRouteSubscribe 2-6
 - publish 3-8
 - publishing 3-9
 - sample of subscription 4-2
 - statements needed to publish 3-9
 - subscription 2-15, 4-2

- tips on using for publishing 3-10
- view subscription 2-15
- where to place 3-9
- PeopleSoft Gateway
 - lookup table 9-6
- PeopleSoft Gateway Servlet 7-3
- PeopleTools
 - messaging objects 2-1
- pinging a remote node 10-14
- printing a message in message monitor 10-26
- properties
 - message channel 2-7
- PSADMIN
 - configuring messaging servers 8-17
 - dedicated message servers 8-2
- PSBRKHND 7-7
- PSBRKSRV 7-7
- PSCAMA
 - audit action codes 3-5, 6-27
 - fields in record 3-4
 - language code 6-26
 - record 3-4
 - table 3-3
- PSMBHND 8-19
 - parameters 8-20
- PSMBSRV 8-19
 - parameters 8-20
- PSPUBHND 7-7
- PSPUBSRV 7-7
- PSSUBHND 7-7
- PSSUBSRV 7-7
- publication
 - defined 7-3
- Publication Broker 7-8, 7-9
 - dedicated 8-4
- publication contract
 - defined 7-3
 - monitoring 10-8
- Publication Contractor 7-8, 7-9
 - dedicated 8-6
- publication ids 6-9
- publish
 - defined 7-3
 - PeopleCode 3-8
 - to a local node 7-11
 - to a third party 7-14
- publish and subscribe
 - across nodes 2-19
- publish/subscribe 7-1
- publishing messages 3-1
 - in various objects 3-8
- purging message tables in message monitor 10-26

Q

- queries page 10-14

R

- reader servlet 9-2
- remote node
 - testing for availability 10-14
- reply document 6-12
 - error codes 6-11
 - format 6-11, 6-12
- return code
 - values 6-13
- routing direction
 - specifying 2-4
- routing rules tab 2-4

S

- servlets
 - in gateway 9-1
- structured message viewer in message monitor 10-23
- subscribe
 - defined 7-3
 - from PeopleSoft 6-15
- subscribing to messages 4-1
- subscription contract
 - defined 7-3
- subscription contract status 5-4
- Subscription Contractor 7-8, 7-9
 - dedicated 8-9
- subscription contracts
 - monitoring 10-9
- subscription PeopleCode 2-15, 4-2

- system pause time for nodes 10-12

T

- test message in application messaging 2-18
- testing
 - application messages 3-10
- testing availability of remote node 10-14

V

- validate incoming data 5-1
- viewing message nodes 2-1
- viewing subscription Peoplecode 2-15

W

- web server
 - application messaging 7-10
 - Application Messaging Gateway 9-3

X

- XML
 - defined 7-3
- XML data
 - and UTF-8 6-9
- XML message viewer 10-23
- XML post 6-43