

HP Client Automation Enterprise

Messaging Server

for the AIX, HP-UX, Linux, Solaris, and Windows® operating systems

Software Version: 7.20

Installation and Configuration Guide

Manufacturing Part Number: none

Document Release Date: August 2008

Software Release Date: July 2008



Legal Notices

Warranty

The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

The information contained herein is subject to change without notice.

Restricted Rights Legend

Confidential computer software. Valid license from HP required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Copyright Notices

© Copyright 1998-2008 Hewlett-Packard Development Company, L.P.

Trademark Notices

Linux is a registered trademark of Linus Torvalds.

Microsoft®, Windows®, and Windows® XP are U.S. registered trademarks of Microsoft Corporation.

OpenLDAP is a registered trademark of the OpenLDAP Foundation.

PREBOOT EXECUTION ENVIRONMENT (PXE) SERVER
Copyright © 1996-1999 Intel Corporation.

TFTP SERVER

Copyright © 1983, 1993

The Regents of the University of California.

OpenLDAP

Copyright 1999-2001 The OpenLDAP Foundation, Redwood City, California, USA.

Portions Copyright © 1992-1996 Regents of the University of Michigan.

OpenSSL License

Copyright © 1998-2001 The OpenSSLProject.

Original SSLeay License

Copyright © 1995-1998 Eric Young (eay@cryptsoft.com)

DHTML Calendar
Copyright Mihai Bazon, 2002, 2003

Documentation Updates

The title page of this document contains the following identifying information:

- Software Version number, which indicates the software version.
 - The number before the period identifies the major release number.
 - The first number after the period identifies the minor release number.
 - The second number after the period represents the minor-minor release number.
- Document Release Date, which changes each time the document is updated.
- Software Release Date, which indicates the release date of this version of the software.

To check for recent updates or to verify that you are using the most recent edition, visit the following URL:

<http://h20230.www2.hp.com/selfsolve/manuals>

This site requires that you register for an HP Passport and sign-in. To register for an HP Passport ID, go to:

<http://h20229.www2.hp.com/passport-registration.html>

Or click the **New users - please register** link on the HP Passport login page.

You will also receive updated or new editions if you subscribe to the appropriate product support service. Contact your HP sales representative for details.

Table 1 indicates changes made to this document.

Table 1 Document Changes

Chapter	Version	Changes
All	7.20	General edits and rebranding from HP Configuration Management to HP Client Automation.
Chapter 1	7.20	Page 12, Using this Guide with Core and Satellite Servers , new topic.
Chapter 2	7.20	Page 24, Platform Support , removed table listing supported platforms; customers should refer to the platform support tables in the HPCA 7.20 Release Notes for that information.
Chapter 2	7.20	Page 26, Installation Procedures for Windows and UNIX : Added warning: For 7.20, it is important to rename core.dda.cfg to core.dda.cfg.old to enable all of the new features, such as thin client support and Vulnerability Management.

Chapter	Version	Changes
Chapter 2	7.20	Page 29, Select the Data Delivery Agents to Install , core.dda is required for this release; Version 7.20 of the core.dda includes embedded support for Windows CE Thin Clients and Vulnerability Management.
Chapter 3	7.20 Aug 2008	Page 41, Configure ODBC Drivers for HP-UX, Linux, or Solaris , corrected the directory where you will find /nvdmodbc when configuring the odbc.ini for unix: Navigate to the /nvdmodbc directory located in your Messaging Server directory.
Chapter 3	7.20	Page 70, About the Sections in the RMS.CFG File . Logging configuration section was updated to reflect updated RMS.CFG.
Chapter 3	7.20	Page 85, Configuring the Log Level, Log Size and Number . Section updated to reflect latest RMS.CFG syntax for logging configuration.

Support

You can visit the HP Software support web site at:

www.hp.com/go/hpsoftwaresupport

This web site provides contact information and details about the products, services, and support that HP Software offers.

HP Software online software support provides customer self-solve capabilities. It provides a fast and efficient way to access interactive technical support tools needed to manage your business. As a valued support customer, you can benefit by using the support site to:

- Search for knowledge documents of interest
- Submit and track support cases and enhancement requests
- Download software patches
- Manage support contracts
- Look up HP support contacts
- Review information about available services
- Enter into discussions with other software customers
- Research and register for software training

Most of the support areas require that you register as an HP Passport user and sign in. Many also require an active support contract. To find more information about support access levels, go to the following URL:

http://h20230.www2.hp.com/new_access_levels.jsp

To register for an HP Passport ID, go to the following URL:

<http://h20229.www2.hp.com/passport-registration.html>

Contents

1	Introduction	11
	Using this Guide with Core and Satellite Servers	12
	About the Messaging Server	12
	Features and Capabilities	12
	Benefits over Previous Implementations.....	13
	Messaging Server Processing on the Configuration Server	13
	About the Data Delivery Agents	17
	About Routing Inventory Data	18
	About the SQL Database Tables and Scripts for Inventory.....	18
	Creating SQL Tables for the Inventory Database	19
	About Using Store and Forward	19
	About this Guide.....	20
	Summary.....	22
2	Installing the Messaging Server	23
	Messaging Server Installation.....	24
	Platform Support.....	24
	Tips.....	24
	Tips for Installing Data Delivery Agents.....	25
	Installation Procedures for Windows and UNIX	26
	Overview of Installation Tasks	26
	Post-Installation Procedures.....	39
	Configure ODBC Drivers for HP-UX, Linux, or Solaris	39
	Set the DBTYPE for a Patch ODBC Database on Oracle.....	43
	Verify the Patch Method Connections and Queue Name	44
	Enable HTTPS Routing using SSL	45
	Revert to a Messaging Server Configuration for Single-queue Processing.....	47
	Configure the USAGE.DDA to Aggregate and Forward Usage Data.....	47
	Starting and Stopping the Messaging Server.....	48

Windows Procedures	48
UNIX Procedures	48
Verify Installation	50
Summary	52

3 Configuring and Tuning the Messaging Server 53

Understanding the Configuration Server Modules that Support the Messaging Server ..54	
Getting Agent information to the Messaging Server	54
About the Patch Method for Collection.....	55
About the ZTASKEND REXX method.....	55
ZTASKEND calls to QMSG	56
Processing Phase-Dependent Objects	56
Processing Always Objects.....	59
Processing under Earlier Versions of Messaging Server: 2.x and 3.x.....	60
QMSG Method Syntax.....	60
How Priority Establishes Messaging Server Processing Order	62
Configuring the Messaging Server	63
Editing the Configuration Files for the Messaging Server and Data Delivery Agents	63
About the Sections in the RMS.CFG File	70
About the Sections in the CORE.DDA.CFG File.....	72
ODBC Settings for CORE, INVENTORY and WBEM Objects	75
About the Sections in the INVENTORY.DDA.CFG File.....	76
About the Sections in the WBEM.DDA.CFG File	79
About the Sections in the PATCH.DDA.CFG File	81
ODBC Settings for PATCH Objects	82
Additional Tuning Topics.....	83
Configuring the Poll Interval and Post Quantity	83
Configuring for Retry Attempts	84
Configuring for Failover	84
Configuring the Log Level, Log Size and Number.....	85
Changing the Logging Level.....	85
Changing the Size and Number of Log Files	86
Configuring the Messaging Server to Discard or Drain Messages	86
Configuring the Messaging Server to Route Portal Messages.....	87
About the Portal Data Queue (rmpq) in CORE.DDA.CFG.....	87

Restoring Routing for Portal Messages.....	88
Disabling Processing of Messages in a Queue.....	89
Modifying the Priority in which Messages are Processed	91
4 Troubleshooting.....	93
Troubleshooting the Messaging Server.....	94
Problem: Log indicates no route defined or failed delivery	94
Solution:.....	94
Problem: Error 404 or 500	94
Solution:.....	95
Summary.....	96
A Optional Messaging Server Configurations.....	97
Example 1: Configuring the Messaging Server for Store and Forward.....	98
Installing and Configuring a "Receiving" Messaging Server	99
About the Messaging Server Receiver	99
Configuring the Receivers for the .dda Modules	100
Running the Installation for a Receiving Server and DDAs.....	101
Configuring a Messaging Server to Forward Messages	102
Forwarding CORE, INVENTORY and WBEM Messages	103
Forwarding PATCH Messages.....	106
Example 2: Configuring the Messaging Server to Route Objects from a Single \Data\Default Queue.....	108
Configuring the Register Default Section	110
Example 3: Configuring Messaging Server to Route to Multiple Queues	111
Example 4: Configuring Data Delivery Agents to Route to Multiple DSNs using ODBC.....	113
Index	117

1 Introduction

At the end of this chapter, you will:

- Know the benefits of the HP Client Automation Messaging Server (Messaging Server).
- Understand Messaging Server processing using message queues and data directory objects.
- Become familiar with the Messaging Server Data Delivery Agent modules.

Using this Guide with Core and Satellite Servers

If your environment uses Core and Satellite servers, first read the Core and Satellite Servers Getting Started Guide as the installation, configuration, and troubleshooting in that guide may override the information in this guide.

About the Messaging Server

The HP Client Automation Messaging Server (Messaging Server) is a robust messaging service that provides a means to forward data from one piece of the HPCA infrastructure to another. It can be used as a point to aggregate different types of data as well as to segregate data accumulated from various HPCA servers by type. Its job is to monitor predefined data queues and dynamically route data objects to one or more external destinations. The Messaging Server provides retry, rerouting, and failover capabilities to ensure all data is transferred efficiently and reliably.

On the HPCA Configuration Server (Configuration Server), the Messaging Server operates hand-in-hand with the executable, QMSG, to handle the transfer of data reported from HPCA agents to the appropriate ODBC reporting database or external HPCA Server.

Features and Capabilities

The Messaging Server provides an efficient and flexible messaging system that can be used by HPCA infrastructure modules. For example, it can:

- Route a single message to multiple destinations.
- Automatically retry a message delivery.
- Re-route messages to a new host after an unsuccessful delivery attempt (failover capability).
- Route data from one Messaging Server to another one (store and forward capability).
- Maintain multiple data queues on Store and Forward Messaging Servers.

Benefits over Previous Implementations

- Multiple, specialized queues allow separate workers to operate on each queue and allows for parallel processing of object messages.
- Additional workers can be configured to drain a queue more quickly.
- Independent data delivery agents allow for modular upgrades.
- Using the Messaging Server to post object data directly to an HPCA Inventory Manager database via ODBC eliminates the need for the legacy Inventory Manager Server.
- Eliminates bottlenecks on the Configuration Server.
- Reliability of processing Inventory and Patch data is maintained, due to:
 - Built-in retry capability.
 - Ability to reroute messages remaining in a queue to a failover location.
 - Retry, holding, and re-routing features eliminate potential loss of data caused by network failures.
- Improved memory control related to creating the Inventory Manager SQL tables and loading the SQL commands upon startup. As a result, these SQL tasks are always performed upon Messaging Server and Data Delivery Agent startup, and support for the previous STARTUPLOAD parameter has been removed.
- Improved queue control and throttling capability for posting HPCA Portal data.
- New support for customized message routing to multiple DSNs using ODBC.

Messaging Server Processing on the Configuration Server

The Messaging Server acts as a delivery service between the Configuration Server and external ODBC databases or HPCA services. It is a separate component from the Configuration Server.

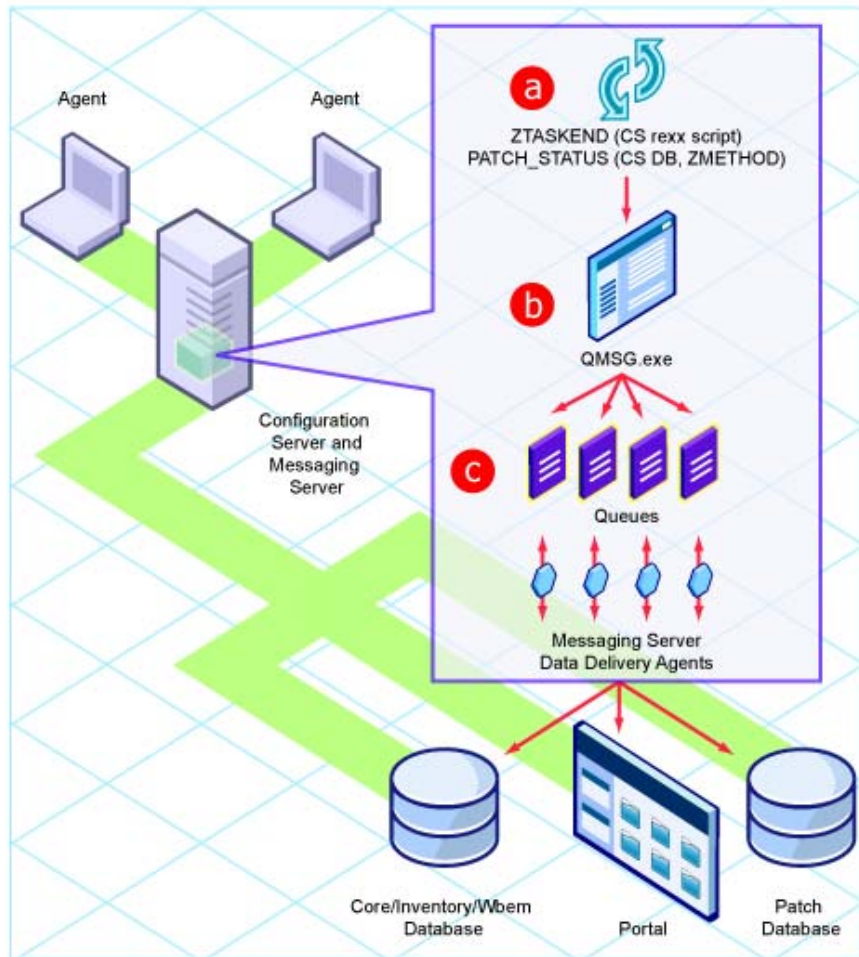
When a customer has multiple Configuration Servers, each one will have a co-located Messaging Server and the ability to route object data to the appropriate target location.

[Figure 1](#) on page 15 provides an overview of Messaging Server Processing.

The HPCA agent connects to the Configuration Server to resolve its desired state. At the end of each agent connection, the agent passes objects back to the Configuration Server. Different agent objects are passed according to each specific phase of the agent connect process.

The Messaging Server refers to CORE objects as the standard HPCA agent objects that are exchanged between the agent and the Configuration Server. Examples of CORE objects are ZMASTER, PREFACE, and SESSION. Other types of objects that can be exchanged are multi-heap WBEM reporting objects, FILEPOST objects created by an inventory agent audit process (called INVENTORY objects) and the DEERROR, BUSTATUS, DESTATUS, RESTATUS and PASTATUS agent objects collected for HPCA Patch Manager processing.

Figure 1 Messaging Server Processing



Legend

- a** ZTASKEND is called
 - b** QMSG.EXE assembles object data
 - c** Messaging Server Data Delivery Agents poll the queues and transfer data
- a** ZTASKEND is Called
On the Configuration Server, the ZTASKEND rexx method is called at the end of the agent connect process. ZTASKEND creates the

commands to invoke the QMSG executable. The commands to QMSG contain parameters that define the appropriate objects to send as well as the designated queues in which to place the objects. ZTASKEND is responsible for naming all objects forwarded to QMSG—with the exception of objects needed for Patch reporting. For Patch objects, five methods in the PRIMARY.SYSTEM.ZMETHOD class of the Configuration Server Database call QMSG for the PATCH objects.

b QMSG.EXE Assemble Object Data

QMSG assembles object data into XML files and creates header files with the appropriate meta data “address” needed to deliver the message by the Messaging Server. When the two message files (XML and meta data files) are created, QMSG places these files in one or more predefined data queues on the Configuration Server.



Prior to Messaging Server version 3.0, QMSG placed all files in a single queue location (that is, the `..\data\default` folder).

For processing efficiencies, QMSG now places objects in separate queues, based on the parameters specified in ZTASKEND and in the five PATCH_* methods (DEERROR, BUSTATUS, DESTATUS, RESTATUS and PASTATUS). The Messaging Server is configured to use individual Data Delivery Agents (DDAs) to process messages from these queues. [Table 2](#) below lists the Data Delivery Agents that operate on each data queue location.

c Messaging Server Data Delivery Agents poll the queues and transfer data

The Messaging Server DDA’s poll the message queues and automatically pick up and transfer data files to the appropriate external locations using the routing type and locations defined in the specific data delivery agent's configuration file.

Table 2 Data Queues and Data Delivery Agents

Data Queue	Data Delivery Agent
<code>..\data\core</code>	<code>core.dda</code>
<code>..\data\inventory</code>	<code>inventory.dda</code>
<code>..\data\wbem</code>	<code>wbem.dda</code>
<code>..\data\patch</code>	<code>patch.dda</code>
<code>..\data\default</code> (<i>prior to MsgSvr v3.0</i>)	<i>none – processed by base MsgSvr</i>


The Messaging Server runs on all Windows and UNIX platforms supported by the Configuration Server.

About the Data Delivery Agents


The Data Delivery Agents are function-specific modules created to transfer certain types of message data. There are Data Delivery Agents available for CORE, INVENTORY, WBEM, and PATCH message data.

- The CORE message data refer to agent objects typically exchanged during a standard agent connect process. Examples of CORE type message objects include ZMASTER, SESSION, and ZCONFIG. CORE objects support reports for Vulnerability Management, Windows CE Thin Clients and Application Management Profiles, among others.
- The INVENTORY message data is comprised of FILEPOST objects created during an agent audit process.
- The WBEM message data is comprised of wbem reporting object data.
- The PATCH message data is comprised of DESTATUS, RESTATUS, BUSTATUS, DEERROR and PASTATUS agent object data.

On all platforms other than AIX, these Data Delivery Agents (DDAs) have the ability to post messages using ODBC into a SQL database that can be used for reporting. For processing efficiency, each DDA works independently on its own queue.

 The Messaging Server's DDAs on AIX platforms do not support ODBC posting to a database. The AIX-supported solution is to reconfigure the DDAs on AIX to forward their messages to a Messaging Server on any non-AIX platform, where the messages can be posted using ODBC. For an overview, refer to [About Using Store and Forward](#) on page 19.

The Messaging Server loads these independent data delivery agents, whose configurations define how and where the messages for CORE, INVENTORY, WBEM, and PATCH data objects will be delivered.

 The CORE data delivery agent is configured to post CORE object data to an Inventory Manager database, as well as CORE object data to a Portal directory.

The data delivery agent modules are located in the `\MessagingServer\modules` folder. The modules are loaded using

“dda.module load” statements in the Messaging Server configuration file (`rms.cfg`).

Each data delivery agent is configured from its own configuration file (`*.dda.cfg`). These configuration files are located in the `\MessagingServer\etc` folder.

About Routing Inventory Data

This Messaging Server supports direct ODBC posting of Inventory Manager data to a back-end SQL Inventory Database (on all platforms other than AIX). The CORE.DDA, INVENTORY.DDA and the WBEM.DDA have the ability to route CORE, INVENTORY and WBEM data messages to a back-end SQL database using ODBC. This is the HP recommended routing option for best performance.

For AIX platforms, the DDAs should be reconfigured to forward messages to another instance of the Messaging Server that is on a non-AIX platform. For an overview, see [About Using Store and Forward](#) on page 19.

About the SQL Database Tables and Scripts for Inventory

The Data Delivery Agents for CORE, WBEM, and INVENTORY post their message data into the same SQL tables as were previously created by the legacy Inventory Manager Server (which is now retired). The default definitions for these tables and associated SQL queries are found in the following Messaging Server directories:

```
/etc/<module name>/sql/hp
```

Customized scripts can be placed in the directories:

```
/etc/<module name>/sql
```


The previous location forces the customized scripts to be loaded and used instead of the ones in the `/etc/<module name>/sql/hp` directories.

The script necessary to map the CORE object data to the related SQL table column is `taskend.tcl`. The script necessary to map the INVENTORY object data (FILEPOST object) is called `filepost.tcl`. Both of these scripts are found in the `/etc/<module name>/lib` directory of the Messaging Server. Any users of the previous Inventory Manager Server can migrate any *customized* scripts directly into the `/sql` directory for the associated Data Delivery Agent module.

Creating SQL Tables for the Inventory Database

On a Windows platform, the SQL tasks to create the tables for the Inventory Database and load the SQL commands into memory are performed when the Messaging Server service and the Data Delivery Agents are initially started.

On HP-UX, Linux or Solaris platforms, you won't see the tables immediately after the installation – these platforms require a post-installation configuration of the ODBC driver. After the ODBC drivers are configured and the Messaging Server is restarted you will see the tables.

 The STARTUPLoad parameter that was available in Version 3.0 to control whether these SQL tasks were performed at startup or upon first use is no longer supported. As of Messaging Server 5.x, all SQL tasks are performed at service startup and any STARTUPLoad value found in a configuration file is ignored.

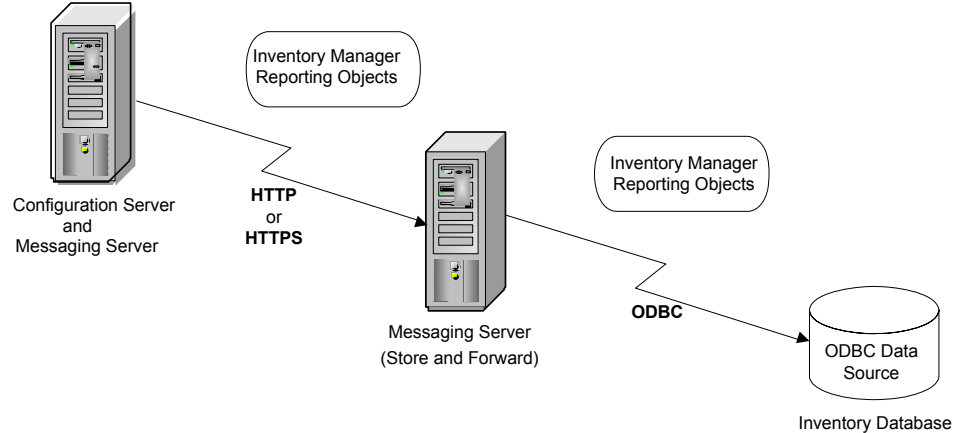
About Using Store and Forward

The Messaging Server includes store and forward capabilities that allow you to forward messages to another Messaging Server using HTTP or HTTPS. This is the AIX-supported solution. In addition, if the Messaging Server is not located close to the SQL database, it is a good practice to forward the messages to one that is located as close to the SQL database as possible.

This version of the Messaging Server supports forwarding and receiving messaging using multiple queues.

Figure 2 on page 20 illustrates a typical configuration that forwards messages to another Messaging Server, prior to posting data to the Inventory database using ODBC.

Figure 2 Store and Forward Messaging Server



- 1 The Messaging Server on the Configuration Server is configured to have the core, inventory, and wbem data delivery agents forward the data to another Messaging Server using HTTP or HTTPS. This configuration makes use of the *coreforward*, *inventoryforward*, and *wbemforward* sections that are provided in the data delivery agent configuration files.
- 2 The Store and Forward Messaging Server is located close to the Inventory Manager SQL database. It receives the core, inventory, and wbem objects (still in separate queues).
- 3 The attending core, inventory, and wbem data delivery agents on the receiving Messaging Server post the data objects to the Inventory Manager database using ODBC.

For more information on this topic, see [Example 1: Configuring the Messaging Server for Store and Forward](#) on page 98.

About this Guide

In addition to this chapter, this book contains the following information:

- **Installing the Messaging Server**
This chapter describes how to install the Messaging Server co-resident with the Configuration Server, and how to start and stop the Messaging Server service.

- **Configuring and Tuning the Messaging Server**
This chapter describes how the Configuration Server ZTASKEND rexx and the QMSG executable work hand-in-hand with the Messaging Server. It also discusses how to configure the Messaging Server configuration file, which loads the Data Delivery Agents. In addition, this chapter also describes how to configure the DDA modules to route CORE, INVENTORY, WBEM, and PATCH message data to the Inventory, Portal, and Patch Manager databases or directories. Additional tuning options are included.
- **Troubleshooting**
This chapter reviews how to resolve common error messages in the `rms.log` files and identifies solutions for typical posting problems.
- **Additional Messaging Server Configuring Options**
This appendix describes alternate configurations, including how to install and configure for Store and Forward, and other customized configurations.
- **Product Name Changes**
This appendix lists the new HP names that have been applied to former Radia products and terms.

Summary

- The Messaging Server routes the object data collected from Client Automation Agents and placed into queues into the appropriate Client Automation server or SQL database. Messages can also be forwarded to another Messaging Server.
- Messages processed include agent objects collected for core, inventory, wbem and patch data.
- Configuration settings in the `rms.cfg` file allow you to load the Data Delivery Agents needed to process the queues on that server. There are separate Data Delivery Agents for core, inventory (filepost), wbem and patch data objects.
- Configuration settings in the `*.dda.cfg` files for the specific data delivery agents specify how and where to route the data processed by that data delivery agent.
- Additional tuning options address load balancing when processing high-volumes of data as well as large-sized objects.

2 Installing the Messaging Server

At the end of this chapter, you will:

- Know how to install the HP Client Automation Messaging Server (Messaging Server).
- Be able to verify the installation of the Messaging Server.



If your environment uses Core and Satellite servers, first read the Core and Satellite Servers Getting Started Guide as the installation, configuration, and troubleshooting information in that guide may override the information in this guide.

Messaging Server Installation

Before you install the Messaging Server, identify the server where the Messaging Server will reside. Among the available choices are the same physical server that is running the Directory Services (or SQL database), the Configuration Server, as well as other remote server locations.

Understanding your network topology as well as the goals of your present network configuration will help you arrive at the best Messaging Server solution. When making the choice of servers for installation of the Messaging Server, please bear in mind the recommended best practice of locating the Messaging Server as close to the SQL database that is receiving the data via ODBC as possible. This solution can be achieved by using multiple Messaging Servers in a Store and Forward configuration. [Configuring a Messaging Server to Forward Messages](#) is discussed in the Appendix, [Optional Messaging Server Configurations](#) on page 102. The Store and Forward capability requires that the Messaging Server is installed and then the configuration file is hand-edited to customize the installation.

Review the reference documentation on the [HP Technical Support](#) Web site to help you determine which machine is best suited in your environment for running the Messaging Server. Install the Messaging Server from the Extended Infrastructure directory on the HPCA infrastructure media

This release supports the post-install configuration of multiple worker processes operating on the same queue. HP recommends starting with the default configuration of a single worker process per queue. If messages are not being processed quickly enough, you can then add another worker to drain a queue more quickly. For more information, refer to [Additional Tuning Topics](#) on page 83.

Platform Support

For the latest and detailed information about supported platforms, refer to the Release Note document that accompanies this release.

Tips

- Review the Release Notes that accompany this release before running the installation for any installation-related adjustments that may apply to your environment.

- Click **Cancel** in any of the windows to exit the installation. If you click **Cancel** accidentally, prompts enable you to return to the installation program.
- Click **Back** at any time to return to previous windows. All the information that you entered thus far will remain unchanged.
- Most windows have associated error messages. If your specifications are invalid, an error message will appear. Click **OK** and enter the correct information.
- This installation program will display recommended default values. Deviation from these default settings must be coordinated with changes in the ZTASKEND rexx. We recommend accepting all defaults for folder names and locations; however, they can be overridden by specifying the parameters necessary to suit your environment.
- The set of prompts to configure either the Messaging Server or the Data Delivery Agents may look similar. Note the configuration file names listed near the top of each window to identify which file is being customized.

RMS . CFG

CORE . DDA . CFG

INVENTORY . DDA . CFG

WBEM . DDA . CFG

PATCH . DDA . CFG

Tips for Installing Data Delivery Agents

- For each Data Delivery Agent you choose to install, additional windows will prompt for the Directory to Scan and routing configuration parameters.
- You can add one or more Data Delivery Agents to an existing Messaging Server install using the same installation program. Once a DDA is installed and its data-specific directory exists, the ZTASKEND method on the Configuration Server automatically redirects the messages to the new directory location.

Installation Procedures for Windows and UNIX



If you have previously installed the Messaging Server, rename the `rms.cfg` file so that a new `rms.cfg` can be created during the install procedure.

To revise the configuration of an existing Data Delivery Agent, rename its existing `*.dda.cfg` file so that a new configuration file can be created during the install procedure. For example, to revise the CORE Data Delivery Agent configuration, rename the existing `core.dda.cfg` file to `core.dda.cfg.old`.

For 7.20, it is important to rename `core.dda.cfg` to `core.dda.cfg.old` to enable all of the new features, such as thin client support and Vulnerability Management.

Overview of Installation Tasks

Because the Messaging Server installation supports prior and current configuration options, there can be many prompts for information during the install that follows. Use [Table 3](#) below as a roadmap to the sequence and contents of the prompts.

Table 3 Messaging Server Installation -- Task Overview

Task	Page	Notes and Tips
1 Launch install, accept license and select the Messaging Server directory	27	Rename an existing <code>rms.cfg</code> file and <code>core.dda.cfg</code> file before you begin.
2 Select Data Delivery Agents to Install	29	Each DDA selected here (except for the <code>usage.dda</code>) adds the following windows: <ul style="list-style-type: none">• Scan directory window in Step 3.• Configuration windows for Tasks 5, 6, 7 and 8. Using Data Delivery Agents is a best practice that improves performance and allows for flexible configurations and easy upgrades. To configure the <code>usage.dda</code> , refer to the <i>HP CM Application Usage Manager User Guide</i> .

Task	Page	Notes and Tips
3 Identify Message Directories to Scan (Message Folder Locations)	30	You are always prompted for the Messaging Server directory to scan, followed by directories to scan for each Data Delivery Agent selected in Task 2. The Patch Directory to Scan must match the <code>ZMTHPRMS -queue</code> value in five <code>PATCH</code> methods. See page 44 for details.
4 Configure the Messaging Server Store and Forward Port	32	Identify a store and forward port the Messaging Server can receive messages on. Default is 3461.
5 Configure the Core Data Delivery Agent (core.dda.cfg)	34	Always displayed – CORE DDA is required. Also routes HPCA Portal objects, if desired.
6 Configure the Inventory Data Delivery Agent (inventory.dda.cfg)	36	Displays if INVENTORY DDA was selected in Step 2. Routes the FILEPOST objects to an Inventory Manager database.
7 Configure the Wbem Data Delivery Agent (wbem.dda.cfg)	37	Displays if WBEM DDA was selected in Step 2.
8 Configure the Patch Data Delivery Agent (patch.dda.cfg)	37	Displays if PATCH DDA was selected in Step 2.
9 Review Installation Summary and Finish	38	Allows review before proceeding with the install.
10 Perform post-installation tasks	39	Important! Review these topics after installation to see which ones apply to your installation. Note: All non-Windows platforms require post-installation tasks.

Task 1 Launch install, accept license and select the Messaging Server directory

- 1 From the HPCA infrastructure media, navigate to the `\extended_infrastructure\messaging_server` directory.
- 2 Navigate to the folder for your operating system.

- 3 On a Windows platform, double-click **setup.exe**

or

On a UNIX platform, enter the following command:

```
./setup
```

and press **Enter**.

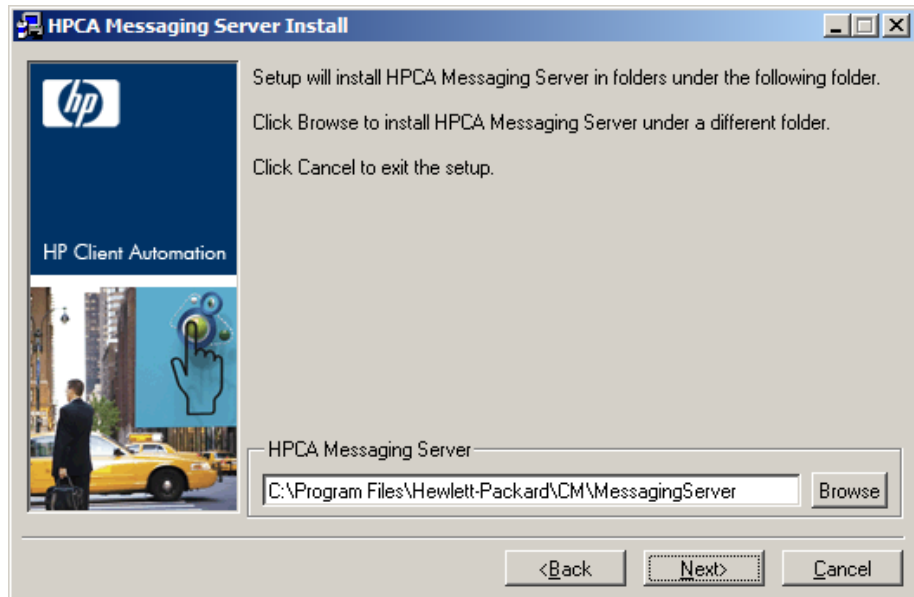
The Welcome window for the Messaging Server Setup program opens.

- 4 Click **Next**.

The End User License Agreement window opens.

- 5 Read the license agreement and click **Accept**.

The Select the installation folder window opens.



- 6 Use this window to select the folder where you want to install the Messaging Server.

- Click **Next** to accept the default installation folder.

or

- Click **Browse** to select a different folder.

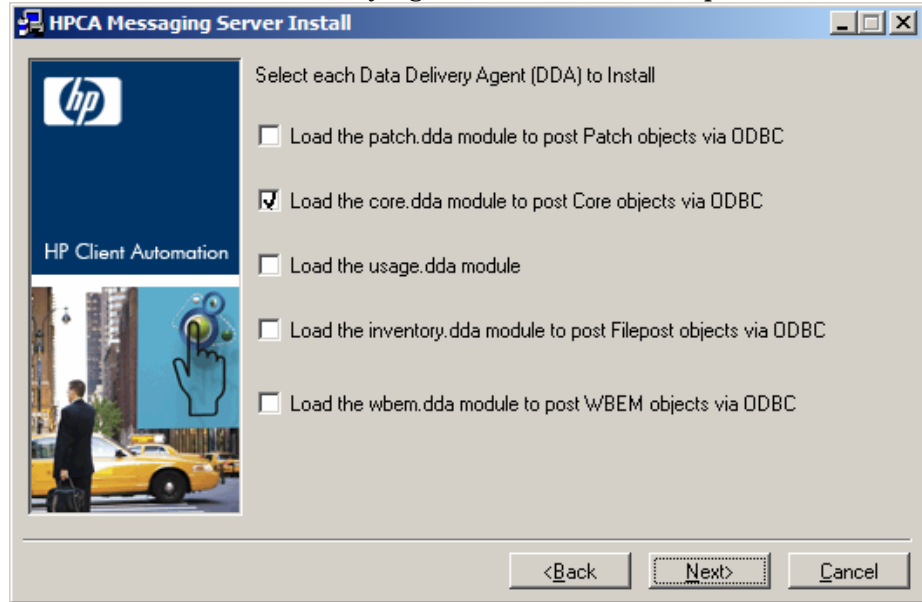
- 7 Click **Next**.

Task 2 Select Data Delivery Agents to Install



The Messaging Server for this version *requires* the use of the DDAs from this version. If you are upgrading your Messaging Server, also upgrade each DDA previously installed.

The Select each Data Delivery Agent to install window opens.



- 8 Use this window to select the check box next to each Data Delivery Agent (DDA) that you want to install on this Messaging Server. See [Table 4](#) below for a list of which DDA(s) to install in order to support a given HP Configuration Management product.

— The usage.dda.module is only available for Windows. Refer to the *HP CM Application Usage Manager User Guide* for post-install configuration.



HP recommends installing Data Delivery Agents for all data objects being collected and reported in your environment.

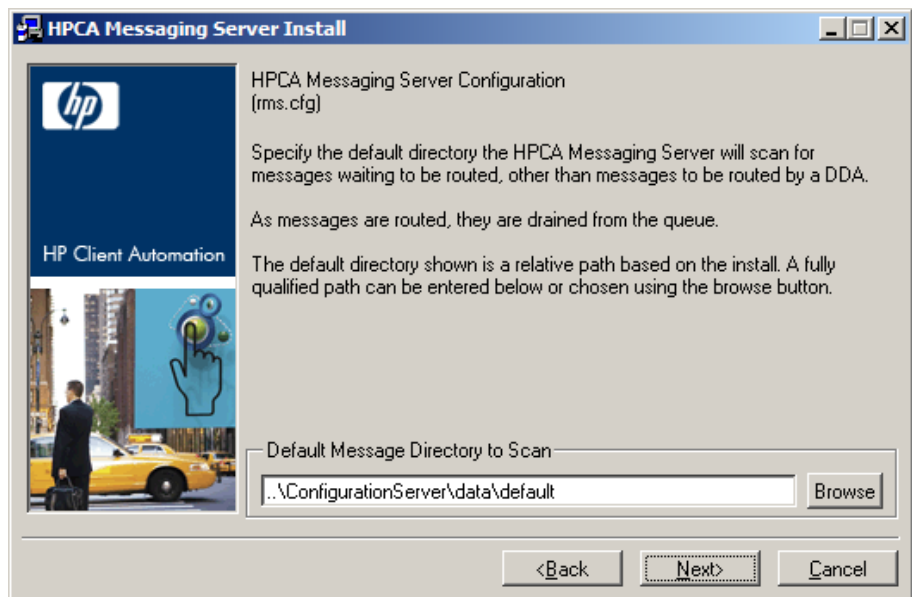
Table 4 DDAs to install by Configuration Management Product

Product	Data Delivery Agents to Install
Application Management Profiles	core.dda

Product	Data Delivery Agents to Install
Application Manager for Windows CE thin clients	core.dda
Application Usage Manager	usage.dda (Windows only)
Inventory Manager	core.dda, inventory.dda and wbem.dda
Portal	core.dda
Patch Manager	patch.dda
Vulnerability Management	core.dda

Task 3 Identify Message Directories to Scan (Message Folder Locations)

The Default Message Directory to Scan window opens.



- 9 Accept the default or click **Browse** to select the directories where the Messaging Server should scan for any messages that are *not* being routed by a Data Delivery Agent. Even if all Data Delivery Agents are installed, this Default Message Directory must exist.

Normally, this is the `\data\default` folder located where the Configuration Server is installed.

This directory is created upon start-up of the Messaging Server if the directory doesn't exist.



If necessary, adjust the directory path to your Configuration Server, but keep the `\data\default` folder names.

10 Click **Next**.

For each Data Delivery Agent that was selected to be loaded, a corresponding Directory to Scan window opens.

Because the Core Data Delivery Agent is required, the Core Data Delivery Agent Configuration window opens and prompts you for the **Core Message Directory to Scan** location..

11 Accept the default location, or click **Browse** to select the directory where the Core Data Delivery Agent should scan for any core messages.

Normally, this is the `\data\core` folders located where the Configuration Server is installed. This directory will be created if it doesn't already exist when the Messaging Server starts. Changes to this directory name have to be coordinated with changes to ZTASKEND REXX for a Messaging Server co-resident with the Configuration Server.



If necessary, adjust the directory path to your Configuration Server, but keep the `\data\core` folder name.

12 Click **Next**.

If the Inventory Data Delivery Agent was selected, the Inventory Delivery Agent Configuration window opens and prompts you for the **Inventory Message Directory to Scan** location.

13 Accept the default location, or click **Browse** to select location where the Inventory Data Delivery Agent should scan for any filepost messages.

Normally, this is the `\data\inventory` folder located where the Configuration Server is installed. This directory will be created if it doesn't already exist when the Messaging Server starts. Changes to this directory name must be coordinated with changes to ZTASKEND REXX for a Messaging Server co-resident with the Configuration Server.



If necessary, adjust the directory path to your Configuration Server, but keep the `\data\inventory` folder name.

14 Click **Next**.

If the Wbem Data Delivery Agent was selected, the Wbem Delivery Agent Configuration window opens and prompts you for the **WBEM Message Directory to Scan** directory.

- 15 Accept the default location, or click **Browse** to select the directory where the Wbem Data Delivery Agent should scan for any wbem messages.

Normally, this is the `\data\wbem` folder located where the Configuration Server is installed. This directory will be created if it doesn't already exist when the Messaging Server starts. Changes to this directory name must be coordinated with changes to ZTASKEND REXX for a Messaging Server co-resident with the Configuration Server.



If necessary, adjust the directory path to your Configuration Server, but keep the `\data\wbem` folder name.

- 16 Click **Next**.

If the Patch Data Delivery Agent was selected, the Patch Delivery Agent Configuration window opens and prompts you for the **Patch Message Directory to Scan** directory.

- 17 Accept the default location, or click **Browse** to select the directory where the Patch Data Delivery Agent should scan for any Patch messages.

Normally, this is the `\data\patch` folder located where the Configuration Server is installed. This directory will be created if it doesn't already exist when the Messaging Server starts up.



If necessary, adjust the directory path to your Configuration Server. HP recommends keeping the `\data\patch` folder name.

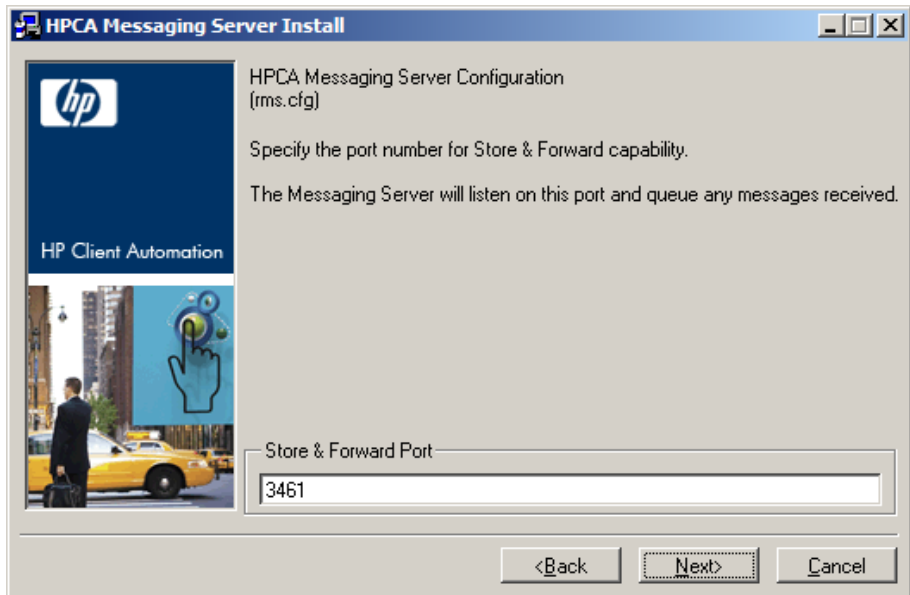


For a Messaging Server co-resident with the Configuration Server, if you enter a directory name other than `patch`, you must modify the `ZMTHPRMS -queue` value in five `PATCH_*` method instances to match. For details, see [Verify the Patch Method Connections and Queue Name](#) on page 44.

The first RMS Configuration window opens.

Task 4 Configure the Messaging Server Store and Forward Port

The Store & Forward Port Setting window opens.



The Messaging Server includes the ability to receive and process messages that have been forwarded from other Messaging Servers in your enterprise. The port used to receive these messages is called the Store & Forward port.

For more information on using store & forward, see [Example 1: Configuring the Messaging Server for Store and Forward](#) on page 98.

- 18 Accept the default store & forward port, 3461, or type another port number to use to receive any messages from other Messaging Servers.
- 19 Click **Next**.

The next window that opens depends upon which DDAs you elected to install.

Task 5 Configure the Core Data Delivery Agent (`core.dda.cfg`)



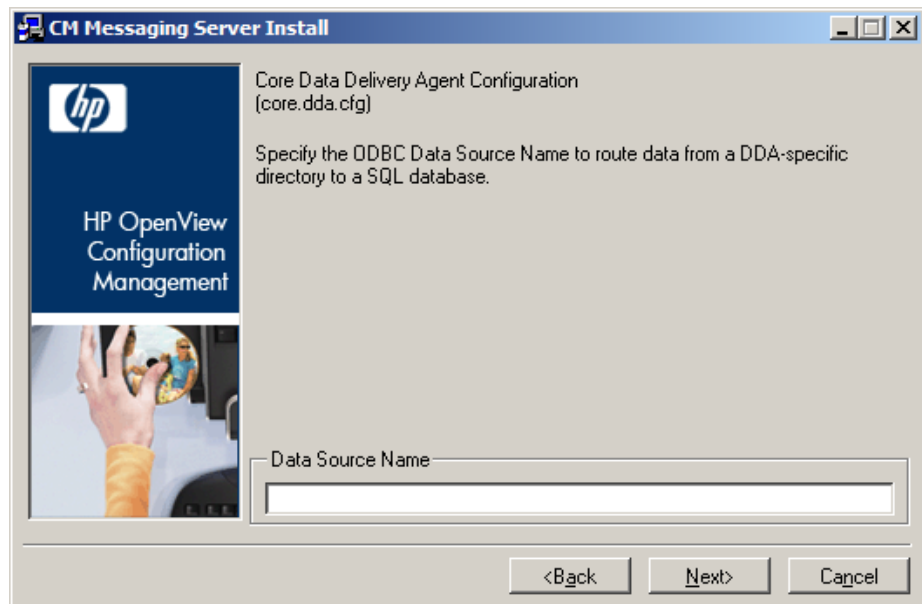
If you elected to load the Core Data Delivery Agent, the five Configuration windows for the Core DDA allow you to specify:

- The ODBC settings (Data Source Name, User Name and Password) to connect to the SQL database for Core message data.
- Optionally, the server and port to post Core data to an HPCA Portal.

If you did not load the Core DDA, these windows do not display.

Specify the ODBC settings to connect to the SQL database for posting Core data in the next three windows.

The Core Data Delivery Agent Configuration (`core.dda.cfg`) window opens for the Data Source Name for posting Core objects.



20 Type the **Data Source Name** of the ODBC SQL database for routing Core message data.

21 Click **Next**.

The Core Data Delivery Agent Configuration (`core.dda.cfg`) window opens for the **DSN User Name** for routing Core data.

22 Type the DSN User Name to use to connect to an ODBC SQL database for Core data.

23 Click **Next**.

The Core Data Delivery Agent Configuration (`core.dda.cfg`) window opens for the **Data Source Password** for routing Core data.

24 Type the password required for the DSN user entered on the previous window.



The password will be encrypted.

25 Click **Next**.

The Core Data Delivery Agent Configuration (`core.dda.cfg`) window opens for the **Portal IP Address**.

26 Type the IP address or DNS host name of the Portal server to route all Portal data found in the Core DDA scan directory location to that server.

or

To automatically discard any Portal data found in the Core DDA scan directory location, leave the Portal IP Address field blank.



The Portal does not require the routing of CORE.RMP data for Wake-On-Lan Notify support.

27 Click **Next**.

The Core Data Delivery Agent Configuration window opens to specify the port of the **Portal Port**.

28 If you entered a Portal IP Address on the previous window, type the port number of the Portal. Normally, this is 3471.

or

Leave the Portal Port field blank if you also left the Portal IP Address field blank.

29 Click **Next**.

Task 6 Configure the Inventory Data Delivery Agent (`inventory.dda.cfg`)



If you elected to load the Inventory Data Delivery Agent, the three Configuration windows for the Inventory DDA allow you to specify the ODBC settings to connect to the SQL Inventory database for posting Filepost objects.

If you did not load the Inventory DDA, these windows do not display.

Specify the ODBC settings to connect to the SQL database for posting Filepost objects in the next three windows.

The Inventory Data Delivery Agent Configuration (`wbem.dda.cfg`) window opens for the Data Source Name for routing Inventory data.

30 Type the **Data Source Name** of the ODBC SQL database for routing INVENTORY message data comprised of Filepost objects for Inventory.

31 Click **Next**.

The Inventory Data Delivery Agent Configuration (`inventory.dda.cfg`) window opens for the **DSN User Name** for routing INVENTORY message data.

32 Type the DSN User Name to use to connect to an ODBC SQL database for routing INVENTORY message data.

33 Click **Next**.

The Inventory Data Delivery Agent Configuration (`inventory.dda.cfg`) window opens for the **Data Source Password** for routing INVENTORY message data.


34 Type the password required for the DSN user entered on the previous window.



The password will be encrypted.

35 Click **Next**.

Task 7 [Configure the Wbem Data Delivery Agent \(wbem.dda.cfg\)](#)

 If you elected to load the Wbem Data Delivery Agent, the three Configuration windows for the Wbem DDA allow you to specify the ODBC settings to connect to the SQL database for posting Wbem data. Many times this is the same SQL database used to post Core and Inventory data objects.

If you did not load the Wbem DDA, these windows do not display.

Specify the ODBC settings to connect to the SQL database for posting Wbem data in the next three windows.

The Wbem Data Delivery Agent Configuration ([wbem.dda.cfg](#)) window opens for the **Data Source Name** for routing Wbem data.

36 Type the Data Source Name of the ODBC SQL database for Wbem data.

37 Click **Next**.


The Wbem Data Delivery Agent Configuration ([wbem.dda.cfg](#)) window opens for the **DSN User Name** for routing Wbem data.

38 Type the DSN User Name to use to connect to an ODBC SQL database for Wbem data.

39 Click **Next**.


The Wbem Data Delivery Agent Configuration ([wbem.dda.cfg](#)) window opens for the **Data Source Password** for routing Wbem data.

40 Type the password required for the DSN user entered on the previous window.

 The password will be encrypted.

41 Click **Next**.

Task 8 [Configure the Patch Data Delivery Agent \(patch.dda.cfg\)](#)

 If you elected to load the Patch Data Delivery Agent, the three Configuration windows for the Patch DDA allow you to specify the ODBC settings to connect to the SQL Patch database (Data Source Name, User Name and Password).

If you did not load the Patch DDA, these windows do not display.

The Patch Configuration window for the Data Source Name for ODBC routing of Patch Data opens. Specify the ODBC settings to connect to the SQL database for Patch on the next three windows.

The Patch Data Delivery Agent Configuration (`patch.dda.cfg`) window opens for the **Data Source Name** for routing Patch data.

42 Type the Data Source Name of the ODBC SQL database for Patch Manager.

43 Click **Next**.

The Patch Data Delivery Agent Configuration (`patch.dda.cfg`) window opens for the **Data Source User Name** for routing Patch data.

44 Type the Data Source User Name to use to connect to an ODBC SQL database for Patch data.

45 Click **Next**.

The Patch Data Delivery Agent Configuration (`patch.dda.cfg`) window opens for the **Data Source Password** for routing Patch data.

46 Type the password required for the DSN user entered on the previous window.



The password will be encrypted.

47 Click **Next**.

Task 9 Review Installation Summary and Finish

After all Data Delivery Agent configurations are completed, the summary of the installation information opens.

48 Click **Install** to begin the installation.

Read and answer any warning dialogs that appear. Which dialog boxes appear will depend on your configuration.

49 Click **Finish** when the installation is finished.

The Messaging Service has been installed and configured for routing data for Inventory Manager, Portal, and Patch Manager, according to your specifications.

Once started, the Messaging Server and any installed Data Delivery Agents scan the various `\data\<queue>` directories on the Configuration Server for message files, and the appropriate data delivery agents deliver the messages to the specified destinations.

The log files for the Messaging Server are placed in the `logs` directory of the `MessagingServer` directory. For example:

`C:\Program Files\Hewlett-Packard\CM\MessagingServer\logs` (on a Windows platform),

or

`/opt/HP/CM/MessagingServer/logs` (on a UNIX platform).



See [Additional Tuning Topics](#) on page 83. Tuning options include changing the polling frequency or retry value, specifying failover servers for inventory data, and adding another Worker process to drain a queue more quickly.

Post-Installation Procedures

Use these procedures to:

- [Configure ODBC Drivers for HP-UX, Linux, or Solaris](#)
- [Set the DBTYPE for a Patch ODBC Database on Oracle](#)
- [Verify the Patch Method Connections and Queue Name](#)
- [Enable HTTPS Routing using SSL](#)
- [Revert to a Messaging Server Configuration for Single-queue Processing](#)
- [Configure the USAGE.DDA to Aggregate and Forward Usage Data](#)

Configure ODBC Drivers for HP-UX, Linux, or Solaris

To install the ODBC Datadirect Connect driver files:

When the Messaging Server is installed on HP-UX, Solaris or Linux platforms, it will copy the following Datadirect Connect driver modules as part of the installation; there is no panel selection needed to install these driver modules:

- `tclobdc` extension needed by the Messaging Server

Target location on all Unix platforms:

`/MessagingServer/modules/nvdodbc.tkd`

- Datadirect 5.2 Connect ODBC drivers

Target location on HP-UX:

/MessagingServer/odbc/nvdmdbc5.20hpux7.tar

Target location on Linux:

/MessagingServer/odbc/nvdmdbc5.20linux7.tar

Target location on Solaris:

/MessagingServer/odbc/nvdmdbc5.20solaris7.tar

When the install completes, the drivers are copied to the locations named above. After installation, the drivers need to be configured for use with the associated database. Configuration involves editing files to set environmental variables, DSN configuration and library locations.

To configure the ODBC drivers:

1 Establish the location of Datadirect libraries:

- a** Copy the .tar file to a different directory if desired such as
/opt/ConnectODBC_5_2
- b** Check and change if necessary the permissions of the
nvdmdbc5.20<platform>7.tar using chmod command.
- c** Extract the contents of the .tar file using the command > tar -xvf
nvdmdbc5.20<platform>7.tar

Extracting the tar creates the nvdmdbc directory.

2 Set Environmental Variables:

The drivers require that several environmental variables be set prior to use. You must have the proper permissions to modify the environmental variables. You should be logged in as a user with these permissions.

The Environmental variable that requires configuration is the Library Search Path.

The library search path variable can be set by executing a shell script located in the nvdmdbc directory.

- a** Locate the shell script nvdmdbc/odbc.sh
- b** Edit the shell script to point to the lib directory where the ODBC libraries are located: ../nvdmdbc/lib
- c** Execute the script to set the environmental variable:
./odbc.sh

Executing the script will set the appropriate library search path environment variable (this is `LD_LIBRARY_PATH` on Solaris and Linux, or `SHLIB_PATH` on HP/UX).

A sample `odbc.sh` looks like:

```
if [ "$SHLIB_PATH" = "" ]; then
    SHLIB_PATH=/opt/odbc/nvdmdbc/lib
else
    SHLIB_PATH=/opt/odbc/nvdmdbc/lib:$SHLIB_PATH
if
export SHLIB_PATH
```

- 3 UNIX and Linux permit the use of a centralized system information file that a system administrator can control. This file contains data source definitions. The file is called `odbc.ini` and located in `nvdmdbc/odbc.ini`. (See "Data Source Configuration" for details). This file can be used to set the environment variable `ODBCINI`, recognized by all DataDirect Connect ODBC drivers, must be set to point to the fully qualified path name of the system information file (`odbc.ini`)

- a Navigate to the `/nvdmdbc` directory located in your Messaging Server directory.
- b Edit the `odbc.ini` for your database connection.
- c Add a DSN to the bottom of the Initial section for the type of database you are connecting to. The example section from the `odbc.ini` shows a connection to a SQL database `SQLTEST`:

```
[ODBC Data Sources]
DB2 Wire Protocol=DataDirect 5.2 DB2 Wire Protocol
dBase=DataDirect 5.2 dBaseFile (*.dbf)
FoxPro3=DataDirect 5.2 dBaseFile (*.dbf)
Informix Wire Protocol=DataDirect 5.2 Informix Wire
Protocol
Informix=DataDirect 5.2 Informix
Oracle Wire Protocol=DataDirect 5.2 Oracle Wire Protocol
Oracle=DataDirect 5.2 Oracle
SQLServer Wire Protocol=DataDirect 5.2 SQL Server Wire
Protocol
Sybase Wire Protocol=DataDirect 5.2 Sybase
Teradata=DataDirect 5.2 Teradata
Text=DataDirect 5.2 TextFile (*.*)
```

```
SQLTEST=DataDirect 5.2 SQL Server Wire Protocol
```

- d Look at the `odbc.ini` for the SQL database type you are connecting to in the different sections. The example will connect to a SQL Server database. Copy the SQL Server section to the bottom of the file for editing. You will configure this copy with the specific information necessary to establish a connection.
- e Change the title to the DSN name, driver location, add the IP address and port where the database resides, database name, logon ID and password. Remove all other values with `< >`

```
[SQLTEST]
Driver=/opt/HP/CM/MessagingServer/odbc/nvdmdbclib/OVmsss2
2.sl
Description=DataDirect 5.2 SQL Server Wire Protocol
Address=QANJ212,1433
AlternateServers=
AnsiNPW=Yes
ConnectionRetryCount=0
ConnectionRetryDelay=3
Database=RIMSQL
LoadBalancing=0
LogonID=HPCA
Password=HPCA
QuotedId=No
SnapshotSerializable=0
```

- f After you have added your DSN to the `odbc.ini` file, you need to set your environment variables in order to load the correct library files for use with your ODBC drivers. Execute the command:

```
ODBCINI=/opt/odbc/odbc.ini;export ODBCINI
```
- g As an alternative, you can choose to make the system information file a hidden file and not set the `ODBCINI` variable. In this case, you would need to rename the file to `.odbc.ini` (to make it a hidden file) and move it to the user's `$HOME` directory.

Note: For Oracle change the default value of `EnableNcharSupport` to 1 to support `nvarchar` datatypes.

- h Check that the new environmental variables have been set running the command: `env`
- i Verify that the load command for `nvdodbc.tkd` is included in the `rms.cfg`.

```
module load nvdodbc
```
- j For HP-UX only: An additional command is needed to enable the dynamic linking of the shared libraries. After running the command:

```
chattr +s enable nvdkit
```

To determine if the dynamic linking of shared libraries is enabled for `nvdkit` run the command:

```
chattr nvdkit
```

The result of running `chattr nvdkit` will show whether the `SHLIB_PATH` is enabled.

Set the DBTYPE for a Patch ODBC Database on Oracle

If you installed the `patch.dda` and the Patch ODBC Database is running on Oracle, you must change the `DBTYPE` parameter in the `patchddaodbc` section of the `patch.dda.cfg` file from `"MSSQL"` to `"ORACLE"`.

To change the `DBTYPE` for `ORACLE`:

- 1 Use a text editor to edit the `patch.dda.cfg` file located in the `\etc` folder of where the Messaging Server was installed.
- 2 Locate the `patchddaodbc` section, and set the `DBTYPE` to `"ORACLE"`. Enclose the value in quotes. An example is shown below:

```
msg::register patchddaodbc {
    TYPE          PATCHODBC
    DSN           "PATCHMGR"
    USER          "PATCH"
    PASS          "<encrypted password>{AES256}"
    DBTYPE        "ORACLE"
```

- 3 Save your changes, and restart the Messaging Server service.

Verify the Patch Method Connections and Queue Name

- HPCA Patch Manager requires five method connections in the Configuration Server Database. For details, refer to the *HPCA Patch Manager Guide*.
- If you installed the `patch.dda` and changed the name of the Patch Message Directory to Scan value during the Messaging Server installation (the expected value is `patch`), you must change the `-queue patch` value in the ZMTHPRMS attribute of the following five PRIMARY.SYSTEM.ZMETHOD instances to match the Patch Directory to Scan value:

PATCH_DEERROR

PATCH_BUSTATUS

PATCH_DESTATUS

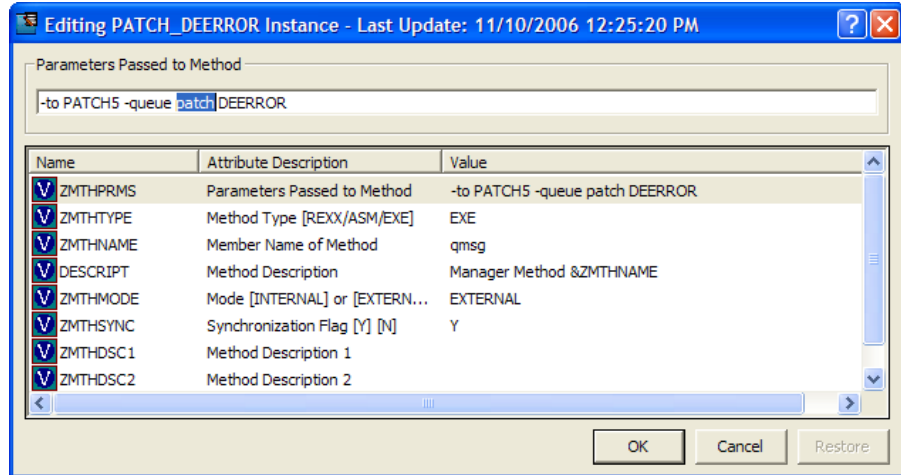
PATCH_RESTATUS

PATCH_PASTATUS

To modify the queue name in the five PATCH_* methods

- 1 Use the HPCA Admin CSDB Editor to edit the ZMTHPRMS attribute of the PRIMARY.SYSTEM.ZMETHOD.PATCH_DEERROR instance, as shown in [Figure 3](#) on page 45.
- 2 Adjust the `-queue patch` value to reflect the directory named as the "Patch Message Directory to Scan".

Figure 3 Specify the Patch queue name in ZMTHPRMS.



For example: if you entered

"`.. \ConfigurationServer\data\mypatch`" as the Patch Directory to Scan for the `patch.dda`, change the value of ZMTHPRMS in the PATCH_DEERROR instance from:

```
-to PATCH5 -queue patch DEERROR
```

to

```
-to PATCH5 -queue mypatch DEERROR
```

- 3 Save your changes.
- 4 Make the same change to the ZMTHPRMS `-queue` value in these PRIMARY.SYSTEM methods:

PATCH_BUSTATUS

PATCH_DESTATUS

PATCH_RESTATUS

PATCH_PASTATUS

- 5 Save your changes.

Enable HTTPS Routing using SSL

Several configuration parameters are required for SSL support in the Messaging Server `RMS.CFG` file. Once these configuration parameters are

available, you can modify the sections in `RMS.CFG` and the various `DDA.CFG` files that route data using HTTP to now route data using HTTPS.

To enable secure HTTPS routing using SSL

- 1 First update your Messaging Server to the latest version.
- 2 Refer to the Messaging Server topic in the *SSL Implementation Guide* for information on how to configure these parameters in the `rms.cfg` file and establish certificates.

```
#-----  
#           RMS SSL Configuration Parameters  
#-----  
Overrides Config {  
    SSL_CERTFILE  "    "  
    SSL_KEYFILE   "    "  
    HTTPS_PORT    "    "  
}
```

The `HTTPS_PORT` is the secure port that the Messaging Server uses to receive messages.

- 3 Edit the sections of the `RMS` or `DDA` configuration files currently defined to route data with a `TYPE` of `HTTP`. Change the `TYPE` from `HTTP` to `HTTPS`, and modify the URL address to include `https:` and the `SSL_port_number` for the server that is receiving the message.

For example, the following entry in the `core.dda.cfg` file routes data to the Portal using `HTTP`:

```
msg::register rmp {  
    TYPE          HTTP  
  
    ADDRESS      {  
        PRI      10  
        URL      http://portal_host:3471/proc/xml/obj  
    }  
}
```

- 4 After enabling SSL, make the following modifications to the same section. These modifications allow for HTTPS routing of HPCA Portal objects to the HPCA Portal service using a secure port of 443.

```
msg::register rmp {  
    TYPE          HTTPS  
  
    ADDRESS      {  
        PRI      10  
        URL      https://portal_host:443/proc/xml/obj  
    }  
}
```

```
}  
}
```

- 5 Apply the same modifications to any other HTTP sections of the RMS or DDA configuration files that you want to route using HTTPS.
- 6 Save the changes, and restart the HPCA Messaging Server service.

Revert to a Messaging Server Configuration for Single-queue Processing

By default, this version of the Messaging Server is configured to include commands to load the Data Delivery Agent (DDA) modules. When these DDA modules are loaded, their associated queue directories are created. The existence of these queue directories is the signal that the data is to be routed directly to an SQL database using ODBC.

To return to the Messaging Server 2.x solution in which messages are placed in a single data queue named `default`, and sent to a legacy CM Inventory Manager Server using HTTP (which then posts them to the SQL database), two post-installation changes are required:

- The Messaging Server configuration file, `rms.cfg`, must be edited to remove the “`dda.module load`” statements for the CORE, INVENTORY and WBEM modules.
- The queue directories created by the DDAs must be removed from the Configuration Server `\data\` directory.

For more information, see [Processing under Earlier Versions of Messaging Server: 2.x and 3.x](#) on page 60, and [Example 2: Configuring the Messaging Server to Route Objects from a Single \Data\Default Queue](#) on page 108.

Configure the USAGE.DDA to Aggregate and Forward Usage Data

Following an installation of the Messaging Server with the `usage.dda`, you must manually configure the `usage.dda.cfg` file to support aggregation and forwarding of usage data files. Refer to the Configuring Aggregation procedures in the *CM Application Usage Manager User Guide*.

Starting and Stopping the Messaging Server

Use the procedures that apply to your type of operating system:

- See the [Windows Procedures](#) below.
- See the [UNIX Procedures](#) below.

Windows Procedures

The Messaging Server is automatically installed as a Windows service. The service name is HPCA Messaging Server.

- Use the Services window of your operating system to start or stop the HPCA Messaging Server.
- Alternatively, to start or stop the installed service from a command prompt, open a DOS window and type the following commands from the `\MessagingServer` directory:

```
nvdkit rms.tkd start
```

```
nvdkit rms.tkd stop
```

- Once the Windows service for the installed Messaging Server is stopped, you can run it from a command prompt. Open a DOS window and type the following command from the `\MessagingServer` directory on your HPCA Configuration Server machine:

```
nvdkit rms.tkd
```

- To stop a Messaging Service running in a DOS window, make the window active and press **Ctrl+C**.

UNIX Procedures

- To start the Messaging Service, go to the `/MessagingServer` directory on your Configuration Server machine and type the following command to run it in the background:

```
./nvdkit rms.tkd &
```

To run the Messaging Service in the foreground, omit the `&` in the previous command.

- To stop the Messaging Service, go to the `/MessagingServer/logs` directory on your Configuration Server machine. First obtain its Process ID (PID) and then kill the process.



The following are general guidelines and the commands are examples that may vary slightly depending on the UNIX type you are using.

- a To obtain the PID for the HPCA Messaging Service, inspect the `rms.pid` file in the `/logs` directory.
- b To stop the service, go to the `/MessagingServer` directory and type:
`nvdkit`
`% kill <PID>`

Verify Installation

Confirm that the Messaging Server is running by performing the following verifications.

- Check the `rms.log` in the `\logs` directory.

- a Look for the entry:

```
Info: HP Client Automation Messaging Service started (PID:
XXXX)
```

This signals that the Messaging Server has started up properly. A sample log entry showing proper startup of the Messaging Service is shown below:

```
Info: -----
Info: HP Client Automation Messaging Service (Version 7.20.000 - Build 229)
Info: Platform: windows XP Service Pack 2 PID: 5812
Info: -----
Info: File C:/Program Files/Hewlett-Packard/CM/MessagingServer/etc/rms.cfg already exists
Info: Loading C:/Program Files/Hewlett-Packard/CM/MessagingServer/etc/rms.cfg...
```

- b Also scan the `rms.log` to note which Data Delivery Agent modules have been loaded. The following sample log entry indicates the DDA module for CORE was loaded:

```
Info: -----
Info: Data Delivery Agent - core.dda - Version 7.20.000 - Build 179
Info: -----
```

- c For each Data Delivery Agent loaded, also scan the `rms.log` to verify the start-up of a worker to process the DDA message queues. For each DDA loaded, look for a worker and its assigned PID. The log entries below show the worker for the CORE Data Delivery Agent is started:

```
Info: -----
Info: Messaging Service - coreq worker - started (PID: 2528)
Info: Platform: Windows_XP Service Pack 2
Info: -----
```

- If the Messaging Server has been started as a Windows service, check that the **HPCA Messaging Server** service has been started in the Services Administrative task section of the Control Panel.
- If the Messaging Server is installed on a Windows platform, check in the Task Manager for the `nvdkit.exe` process. If installed on a UNIX platform, check for the `nvdkit` processes running on UNIX.

The Messaging Server will start a single main `nvdkit` process and a separate `nvdkit` process for each worker process. Each DDA module installed will be a separate worker process and the Messaging Server base module also has its own worker process. Therefore, if you have installed these four DDA modules (`core`, `wbem`, `inventory` and `patch`), there should be six `nvdkit` processes started for the Messaging Server.

Summary

- Understand your network topology and have the targets for the Messaging Server routes laid out before installing the Messaging Server.
- To create a Messaging Server environment on your Configuration Server, the Configuration Server must include a version of ZTASKEND REXX method that calls the QMSG executable for Inventory Manager and Portal data objects. To route Patch Manager objects, the Configuration Server must have a method connection to five PATCH_* instances in the PRIMARY.SYSTEM.ZMETHOD class.
- The Messaging Server is installed as a Windows service or a UNIX process.
- The Usage Data Delivery Agent must be configured manually after the installation. Refer to the *CM Application Usage Manager User Guide*.
- Verify installation by checking that the Messaging Service is running, it is loading the selected Data Delivery Agent modules, and there is a worker process started for both the Messaging Server and each DDA that was loaded.

3 Configuring and Tuning the Messaging Server

At the end of this chapter, you will:

- Be able to understand the Configuration Server methods that support the Messaging Server.
- Be able to configure the parameters in `rms.cfg`.
- Be able to configure the parameters in the core, inventory, wbem and patch data delivery agent files (`core.dda.cfg`, `inventory.dda.cfg`, `wbem.dda.cfg` and `patch.dda.cfg`, respectively).
- Be able to configure the Messaging Server for failover.



- The first part of this chapter discusses the Configuration Server modules that support the Messaging Server.
- The topics on configuring and tuning the Messaging Server and Data Delivery Agents begin on page 63.
- To configure a Messaging Server for Store and Forward, see [Example 1: Configuring the Messaging Server for Store and Forward](#) on page 98.
- To configure a Messaging Server Data Delivery Agent to post messages to multiple DSNs, see [Example 4: Configuring Data Delivery Agents to Route to Multiple DSNs using ODBC](#) on page 113.
- To configure the Usage Data Delivery Agent, refer to the *CM Application Usage Manager User Guide*.

Understanding the Configuration Server Modules that Support the Messaging Server

This topic explains how the methods on the Configuration Server work hand-in-hand with the Messaging Server to collect, queue, and then deliver data to the appropriate external location.

Getting Agent information to the Messaging Server

Agent objects exchanged with or created on the Configuration Server during an agent connect session are formatted into messages for the Messaging Server via the Configuration Server binary executable QMSG. The QMSG executable is part of the standard CM-CS release. This executable is invoked during agent taskend processing by the rexx method ZTASKEND or a connection to the Patch Manager methods. (Refer to [Verify the Patch Method Connections and Queue Name](#) on page 44 for details on the Patch Manager methods. The calls to QMSG can include parameters specifying what queue to place the messages in, the priority in which the message is to be processed, the objects that are to be included in the messages, and the “destination address” or routing identifier for the file.

The QMSG executable formats the object data into an XML file. Each XML file can be made up of multiple objects, such as when processing the CORE objects (for example, ZMASTER, SESSION and ZCONFIG) or it can be a single multi-heap object such as a wbem or filepost object. Each call to QMSG will produce two files, the XML file created from the object data and a file which contains the meta data. Meta data are attributes describing the XML file, how big it is, when it was created and the routing identifier. The actual file names are created with a timestamp format. This enables the Messaging Server to process the oldest messages first. The Messaging Server always processes in a “First In First Out” mode when the messages have the same processing priority.

When queue designations are specified when invoking QMSG, messages are placed in a directory with the specified queue name. When no queue identifier is used, messages are placed in a directory named default. Each data delivery agent uses its own unique queue for its particular messages. This segregation of messages according to type allows for the simultaneous processing of all queues and leads to more efficient operation.

In Summary:

- For agent information needed by the Patch Manager, QMSG is executed as a result of the `SYSTEM.ZMETHOD.PATCH*` instances: `PATCH_DEERROR`, `PATCH_BUSTATUS`, `PATCH_DESTATUS`, `PATCH_RESTATUS` and `PATCH_PASTATUS`.

Refer to the *HPCA Patch Manager Installation and Configuration Guide (Patch Manager Guide)* for more information on creating the method connection needed for Patch message processing. The format of the QMSG call is included on page 65.

- For information needed by the Inventory Manager, Portal, and Application Management Profiles, the Configuration Server REXX method, `ZTASKEND`, is used to trigger the call to QMSG. The format of the QMSG call is included in the discussion of `ZTASKEND` which follows.

About the Patch Method for Collection

Five methods call the QMSG executable with the parameters in the `ZMTHPRMS` attribute of the method. The default value of this attribute looks like this:

```
ZMTHPRMS      -to PATCH5 -queue patch <<object-name>>
```

The `-to PATCH5` parameter specifies that the messages will be placed in a queue called `./data/patch` relative to the location of where QMSG is executed. This is the default location specified in the install of the `patch.dda`.

The `<<object-name>>` parameters (`DEERROR`, `BUSTATUS`, `DESTATUS`, `RESTATUS` and `PASTATUS`) specify the objects that will be included in the message files created by QMSG.

About the ZTASKEND REXX method

The `ZTASKEND` REXX method on the Configuration Server is called at the end of each agent connect, while objects associated with the present session are still available in storage on the CM-CS. `ZTASKEND` invokes `QMSG` when agent data needs to be collected for another service, such as Inventory Manager, Application Management Profiles, or the Portal. QMSG collects the data and places the messages in specified queues for pickup and processing by the Messaging Server and its data delivery agents.



As of `ZTASKEND v 1.9`, different object types (core, inventory and `wbem` objects) are placed in separate queues whenever the Data Delivery Agents have been installed for those data objects. Previous

versions of ZTASKEND placed all data objects in a single queue (named `/data/default`).

An important job of ZTASKEND is to ensure unique agent data is collected at the appropriate agent connect phase. For efficiency, ZTASKEND also groups identical messages, having the exact same object content, to minimize the number of calls made to QMSG.

This topic explains:

- How ZTASKEND determines when to call QMSG for the various agent connect phases and which objects are collected.
- The basic syntax of calls to QMSG.
- How the QMSG `-to` parameter establishes one or more destinations for message processing.
- How the QMSG `-priority` parameter establishes Messaging Server processing order.

ZTASKEND calls to QMSG



This topic reflects the ZTASKEND v1.9 (or above) method delivered with the Radia 4.x releases and Messaging Server versions 3.x.

Processing Phase-Dependent Objects

Each time an HPCA agent connects to the Configuration Server, the agent declares its connection intent or phase. An important role of the ZTASKEND rexx code is to minimize the collection of duplicate information. With that goal in mind, the ZTASKEND method invokes QMSG depending on the agent connection phase and the objects present. ZTASKEND restricts message posting to five specific connection phases. The phases of interest are:

- `BOOTSTRAP` (Client Operations Profiles or COP)
- `AGENT SELF MAINTENANCE`
- `CATALOG RESOLUTION`
- `SERVICE RESOLUTION`
- `AGENT REPORTING`

Processing CORE Objects by Phase

There are "critical objects" collected for each of the core targets for Inventory Manager (RIM) and Portal (RMP). The potential critical objects are:

For RIM: APPEVENT MSIEVENT SYNOPSIS RNPEVENT

For RMP: SYNOPSIS

In addition to the above critical objects, [Table 5](#) below identifies critical objects collected for each phase.

Table 5 Critical Objects collected by phase

Phase	Critical Objects
BOOTSTRAP (COP)	SESSION PREFACE ZSTATUS SMINFO
AGENT SELFMAINTENANCE	SESSION PREFACE ZSTATUS SMINFO
CATALOG RESOLUTION	SESSION PREFACE ZSTATUS ZCONFIG ZMASTER SMINFO
SERVICE RESOLUTION	SESSION PREFACE ZSTATUS SMINFO
AGENT_REPORTING	SESSION PREFACE ZSTATUS SAPSTATS ZRSTATE SMINFO

With this in mind, ZTASKEND uses the following logic:

- 1 Whenever a critical object is presented, the critical object and the additional objects are processed by QMSG and deposited into queues for processing by RMS.
- 2 If a critical object is not present, then the code invokes QMSG when an agent connects during the phases CATALOG_RESO and AGENT_REPORTING.
- 3 If a critical object is not found, then code does not invoke QMSG for the following agent phases: BOOTSTRAP (COP), SERVICE RESOLUTION and AGENT_SELF MAINTAINANCE.
- 4 Finally, ZTASKEND does not invoke QMSG to obtain error message objects (ZERRORM & ZERROR).

[Table 6](#) on page 58 summarizes the Agent Connect phases and the objects collected during the ZTASKEND calls to QMSG, for CORE data going to Inventory Manager or the Portal.

Table 6 ZTASKEND calls to QMSG for CORE Data for RIM and RMP

Agent Connect Phase	QMSG call if not critical object?	QMSG call if critical object
<p>Bootstrap - COP Resolution for Client Operations Profile.</p>	<p>No</p>	<p>Collects these objects for CORE.RIM and CORE.RMP destinations: APPEVENT MSIEVENT SYNOPSIS RNPEVENT SESSION PREFACE ZSTATUS SMINFO</p>
<p>Agent Maintenance Phase</p>	<p>No</p>	<p>Collects these objects for CORE.RIM and CORE.RMP destinations: APPEVENT MSIEVENT SYNOPSIS RNPEVENT SESSION PREFACE ZSTATUS SMINFO</p>
<p>Catalog Resolution: Agent connects to the Configuration Server to obtain service resolution list.</p>	<p>Always. Collects these objects for CORE.RIM and CORE.RMP destinations: SESSION PREFACE ZCONFIG ZMASTER ZSTATUS SMINFO</p>	<p>See previous column, but also collects APPEVENT MSIEVENT SYNOPSIS RNPEVENT.</p>
<p>Single Service Resolution: For each service to be resolved, agent makes another connection to the Configuration Server.</p>	<p>No</p>	<p>Collects the following objects for CORE.RIM and CORE.RMP destinations: APPEVENT MSIEVENT SYNOPSIS RNPEVENT SESSION PREFACE ZSTATUS SMINFO</p>

Agent Connect Phase	QMSG call if not critical object?	QMSG call if critical object
Agent Reporting Phase: At the end of service resolution. Agent data is reported back to the Configuration Server.	Always. Collects these objects for CORE.RIM destination: SESSION PREFACE ZSTATUS SAPSTATS ZRSTATE SMINFO	See previous column, but also collects APPEVENT MSIEVENT SYNOPSIS RNPEVENT

Adding Items to a Critical Object List

The ZTASKEND rexx code can be configured to add object names to the critical and additional object lists. The rexx variables `CriticalRIMObjects` and `CriticalRMPObjcts` contain the object names for each of these targets. The rexx function call to `BuildObjectList` is used to build the object list for each phase.

```

Call BuildObjectList ....
:
:
:
:
CriticalRIMObjects    = "APPEVENT MSIEVENT SYNOPSIS RNPEVENT"
CriticalRMPObjcts    = "SYNOPSIS"

```

The ZTASKEND rexx code contains additional information on how to alter these items.

Processing Always Objects

There is a section of the ZTASKEND rexx code to define objects that will always be processed, independent of the phase being processed. The larger Inventory Manager objects, FILEPOST, WBEMAUDT and CLISTATS are processed this way. If these objects exist, then they are sent to the specified target. In addition to the larger Inventory Manager objects, the job objects: JOBSTAT, JOBPARM, and JOBTASK are also processed this way.

Adding Custom Objects to the BuildAlways Object List

This part of the rexx code can also be configured to add "custom" objects that are always delivered to the specified target. The rexx function `BuildAlways` is used to configure the target, queue and objects to process. The rexx code contains additional information on how to alter these items.

```
Call BuildAlways "inventory", InventoryQueue, "FILEPOST"
```

Processing under Earlier Versions of Messaging Server: 2.x and 3.x

The ZTASKEND rexx code is aware of and supports Messaging Server versions 2.x and version 3.x. It does this by checking to see if (data) queues other than "default" exists. With Messaging Server version 2.x, all posting was done to just one queue, named "default". Messaging Server version 3.x introduces additional queues for each data delivery agent, including "core", "inventory" and "wbem".

Since Messaging Server version 2.x posts all objects to one queue, it uses the `-priority` switch of the `QMSG` call syntax to defer processing of larger Inventory Manager objects. With Messaging Server 3.x, multiple "queues" are used to control message processing and the `-priority` switch of the `QMSG` call is not necessary.

QMSG Method Syntax

The `QMSG` command/method is used to post Configuration Server objects for Inventory Manager, Portal and Application Management Profiles. `QMSG` reads the specified object and converts it to XML and then writes it to the specified queue.

The syntax of the `QMSG` method is given below:

```
qmsg -to <destination(s)> -queue <queue> -priority <priority> object1 object2 ... objectn
```

-to <destination(s)>

must be explicitly coded with one or more destinations, or targets. Messages going to multiple destinations have comma-separated entries. For example:

```
-to CORE.RIM,CORE.RMP
```

The Messaging Server version 2.x destination values used by the delivered `QMSG` include:

```
-to CORE.RIM
```

```
-to CORE.RIM,CORE.RMP (these messages are delivered to both destinations)
```

```
-to INVENTORY
-to INVENTORY.WBEM
```

The Messaging Server version 3.x destination values used by the delivered QMSG include:

```
-to CORE.ODBC
-to CORE.RMP
-to INVENTORY.ODBC
-to WBEM.ODBC
```

For Messaging Server version 3.x and 5.x processing, each message destination requires an equivalent ROUTE defined for it in the msg::register router section of the appropriate Data Delivery Agent's *.dda.cfg file. [Table 7](#) below gives the destination values of QMSG and the configuration file and section used to define its ROUTE.

Table 7 QMSG Destinations and DDA Configuration Locations

QMSG -to destination	Configuration File in <i>MessagingServer\etc</i> folder	Required ROUTE section
-to CORE.ODBC	core.dda.cfg	msg::register corerouter
-to CORE.RMP	core.dda.cfg	msg::register corerouter
-to INVENTORY.ODBC	inventory.dda.cfg	msg::register inventoryrouter
-to WBEM.ODBC	wbem.dda.cfg	msg::register wbemrouter

The version 2.x destination values used by the delivered QMSG include:

```
-to CORE.RIM
-to CORE.RIM,CORE.RMP (these messages are delivered to both
destinations)
-to INVENTORY
-to INVENTORY.WBEM
```

For version 2.x processing, each message destination requires an equivalent ROUTE defined for it in the msg::register router section of the rms.cfg file, as discussed in Example 2: Configuring the Messaging Server to Route Objects from a Single \Data\Default Queue on page 108.

-queue <queue>

The queue is a directory relative to where QMSG.EXE exists. QMSG is located in the \bin directory of the CM-CS. For example, on a Windows platform, if

QMSG resides at

```
C:\CM\ConfigurationServer\bin\qmsg.exe
```

Then the queues would reside at:

```
C:\CM\ConfigurationServer\data\blue
```

```
C:\CM\ConfigurationServer\data\default
```

```
C:\CM\ConfigurationServer\data\green
```

```
C:\CM\ConfigurationServer\data\red
```

The parent directory of all queues is "data". For illustrative purposes, this example shows the existence of the (fictitious) blue, green and red queues. Note that the queue named "default" is the default queue.



Queue locations are defined near the top of ZTASKEND.

For Patch routing, the queue location is named in the parameters passed to QMSG from the five `SYSTEM.ZMETHOD.PATCH_*` instances. The instances are named DEERROR, BUSTATUS, DESTATUS, RESTATUS and PASTATUS. For example:

```
Method = qmsg
```

```
Parameter = -to PATCH5 -queue patch <<object>>
```

To modify the parameters to pass, edit the value of the `ZMTHPRMS` attribute in the `PATCH_*` instance.

-priority

is available to establish Messaging Server processing priority. If omitted, a default priority of 10 is given to the message. Valid values are 00 (highest priority) to 99 (lowest priority). For more information on Messaging Server processing priority, see the topic [How Priority Establishes Messaging Server Processing Order](#) below.

object1 [objectn]

The rest of the command line includes the names of the objects to queue, object1 object2... objectn. The objects are processed in the order specified; thus, depending on the destination, there might be a dependent order.

How Priority Establishes Messaging Server Processing Order

When ZTASKEND calls QMSG, the optional **-priority** parameter in the call assigns a processing priority to the message. Priority values can range from 00 to 99, with 00 reserved for critical processing and 99 being the lowest priority.

For messages waiting to be processed in the same queue, the Messaging Server processes all messages assigned to a higher priority (such as 10) *before*

processing any messages assigned to a lower priority (such as 20). Within a given priority, messages are processed using first in, first out (FIFO) order.



The message priority remains the same for the life of the message. For example, if a message is forwarded from one Messaging Server to another, the message priority remains the same.

- Priority 10 is the default if a priority is not specified.
- Previously, ZTASKEND called QMSG with parameters to assign a lower priority of 20 to the larger objects collected for Inventory Manager Reports: these include file audits, wbem reporting data, and agent statistics (CLISTATS).
- This is no longer necessary because of the segregation of queues by object type.

If the Messaging Server is not able to process the messages as fast as they are delivered from QMSG, the lower priority messages will accumulate at the bottom of a queue location, even though newer messages with higher priorities are still being processed.

Configuring the Messaging Server

Use these topics to reconfigure or tune the Messaging Server after installation, or reconfigure or tune the data delivery agents for core, inventory, wbem or patch data.

Editing the Configuration Files for the Messaging Server and Data Delivery Agents

The Messaging Server and Data Delivery Agents standard installation allows configuration of several of the configuration parameters contained in the respective configuration files. You will need to edit the configuration file with a text editor to achieve a more customized environment.

All of the configuration files for the Messaging Server and Data Delivery Agents are found in the `\etc` directory of where the Messaging Server was installed.

All the configuration files for the Messaging Server and the associated Data Delivery Agents have similar configuration sections. Understanding these

sections and the syntax used to configure them will aid in customizing your environment.

The structure for the RMS configuration file sections is given below:

```
msg::register <unique identifier> {  
    TYPE          <RMS registered TYPE>  
    <Configurable Variable for the registered TYPE>      <Value for that Variable>  
    <Configurable Variable for the registered TYPE>      <Value for that Variable> ...  
}
```

Each configuration section starts with the command `msg::register`. This signals the start of a configuration parameter for the Messaging Server and its modules.

A unique identifier follows the `msg::register` command. Within an instance of the Messaging Server, which includes the configuration files for the Messaging Server and all of the DDA modules, this unique identifier label can only be used once. The unique identifier is followed by a curly brace "{". The configuration section for this TYPE must be ended by a closing curly brace for the entire configuration to work.

All configuration file sections have a TYPE identifier, which indicate the kind of work to be done by this section. These are the current acceptable TYPE designations:

- QUEUE
- ROUTER
- HTTP
- HTTPS
- HTTPD
- FILTER
- ODBC
- COREODBC (only configurable in `core.dda.cfg` and `inventory.dda.cfg`)
- WBEMODBC (only configurable in `wbem.dda.cfg`)
- PATCHDDAODBC (only configurable in `patch.dda.cfg`)
- PATCHDDASUMMARIZE (only in `patch.dda.cfg`)


[Table 8](#) on page 65 describes the different section TYPES and their configurable parameters. The sections are listed in the general order in which they appear in the configuration files.

Table 8 Glossary of Section TYPEs and Configurable Parameters

Section TYPE	Configurable Parameters
<p>QUEUE Defines the directory where messages are placed for processing.</p>	<p>DIR The directory name of the queue.</p> <p>USE Specify the name of the unique identifier that will signify how to dispatch the message. Usually this is a ROUTER type.</p> <p>POLL By default, the Messaging Server is configured to poll the queue location every 10 seconds and post up to 100 objects at a time. To change the poll interval, modify the POLL parameter.</p> <p>COUNT Maximum number of objects to post at a time (during the polling interval defined by POLL). Default is 100 objects.</p> <p>ATTEMPTS Maximum number of attempts to retry a failed message delivery before discarding the message. Default is 200 attempts. (By default, the Messaging Server and Data Delivery Agents are configured to retry any failed posts every hour, and make up to 200 attempts.)</p> <p>DELAY Number of seconds to wait between attempts to retry a failed message delivery. Default is 3600 seconds, or one hour.</p> <p>Note: To calculate the maximum time that a message could stay in the queue before being discarded, take the DELAY time and multiply it by the ATTEMPTS value. Using the default settings, this is a DELAY of 3600 seconds x 200 ATTEMPTS, or approximately eight days.</p>
<p>ROUTER Defines where the messages are going to sent.</p>	<p>ROUTE Delimit each Route by curly braces</p> <p>TO This is the address of the message. It is contained in the meta data file of each message when the message is created.</p>

Section TYPE	Configurable Parameters
	<p>USE Specify the unique name of a Messaging Server TYPE that will be used to dispatch the message, such as HTTP, HTTPS or ODBC. (In a DDA file, the USE entries may also be COREODBC, WBEMODBC and PATCHDDAODBC).</p>
<p>HTTP This is a way to post the message to another server location using http protocol. The target server can be another Messaging Server where the message can be re-queued or it can be another part of the infrastructure, such as the Portal.</p>	<p>ADDRESS Delimit each address using curly braces. Multiple URL destinations can be specified within each HTTP TYPE as long as each has its own ADDRESS label and a different priority.</p> <p>PRI Denotes the priority in which to sent messages to the companion URL. The default priority is 10. The priority setting only matters if multiple ADDRESS entries are configured. If multiple ADDRESS entries are configured the lowest priority is tried first and if that fails the next priority URL is tried. This allows failover capability if there are network problems.</p> <p>URL Specifies the URL to use to send the message.</p>
<p>HTTPS This is a way to post the message using a secure socket. The HTTPS parameters are configured the same as the HTTP parameters--with the exception of the URL specification. The URL uses the https:// convention.</p>	<p>ADDRESS Same as TYPE of HTTP.</p> <p>PRI Same as TYPE of HTTP.</p> <p>URL Specifies the URL using the https:// convention.</p> <p>Refer to the <i>HP Configuration Server SSL Implementation Guide</i> for details.</p>
<p>HTTPD Defines the parameters the Messaging Server will use to receive incoming messages.</p>	<p>PORT Defines the Port used to “listen” for messages. Only one port specification can be used for a given Messaging Server.</p> <p>URLHANDLER Delimit with curly braces. If the incoming messages are to be deposited into different queues, depending upon the URL, the URLHANDLER must be used to</p>

Section TYPE	Configurable Parameters
	<p>delimit the USE and URL parameters.</p> <p>USE Specify the name of the QUEUE type that will receive the incoming messages.</p> <p>URL Specify the URL prefix that that will be accepted by the Messaging Server. When messages are received with the designated URL they are deposited in the associated queue. The QUEUE type must be defined when messages are received. All URL's specified must start with /proc/.</p>
<p>ODBC Used to post PATCH messages into a SQL database.</p>	<p>DSN Data Source Name</p> <p>USER User ID for the DSN</p> <p>PASS Password for the DSN</p>
<p>FILTER A means to route PATCH data into multiple SQL tables.</p>	<p>USE Specify the name of the unique identifier that will signify how to dispatch the message.</p> <p>TO This is the address of the message. It is contained in the meta data file of each message when the message is created.</p> <p>FILTER Used for PATCH object processing into multiple tables</p>
<p>COREODBC Used to post CORE messages into a SQL database. Only configurable in <code>core.dda.cfg</code> and <code>inventory.dda.cfg</code>.</p>	<p>DSN Data Source Name</p> <p>USER User ID for the DSN</p> <p>PASS Password for the DSN</p> <p>DSN_ATTEMPTS Number of attempts to connect to the DSN. Default is 1.</p> <p>DSN_DELAY Delay in seconds between attempts to connect to the</p>

Section TYPE	Configurable Parameters
	<p>DSN. Default is 5 seconds.</p> <p>DSN_PING Delay in seconds between pinging the database connection to verify the DSN is available. Default is 300 seconds.</p> <p>STARTUPLoad No longer supported; any existing value is ignored.</p> <p> As of Version 5.0, the SQL tasks to create the tables and load the scripts for the Inventory Database are always performed upon Messaging Server and Data Delivery Agent startup.</p>
<p>WBEMODBC Used to post WBEM data messages into a SQL database. Only configurable in <code>wbem.dda.cfg</code>.</p>	<p>See COREODBC for common parameters.</p> <p>AUTOCREATE A switch to enable the creation of a new SQL file and table in the Inventory ODBC database when a new object class is received. Default is 0.</p> <p>0 – Does not create a SQL file or table entry for a new object class.</p> <p>1 – Creates a new SQL file and table entry for a new object class.</p>
<p>PATCHDDAODBC Used to post PATCH data messages into a SQL or Oracle database. Only configurable in <code>patch.dda.cfg</code></p>	<p>DSN Data Source Name</p> <p>USER User ID for the DSN</p> <p>PASS Password for the DSN</p> <p>DBTYPE Database type of MSSQL (for Microsoft SQL Database) or ORACLE (for an Oracle Database).</p>
<p>PATCHDDASUMMARIZE Translates ZOBJSTAT objects from pre-5.0 patch agents into the objects used in Version 5.0 patch reporting. Only in</p>	<p>No configurable parameters.</p>

Section TYPE	Configurable Parameters
patch.dda.cfg.	

You can adjust default values and routing options by editing the *.cfg files, located in the \etc directory of where the Messaging Server was installed.

The base Messaging Server configuration file (rms.cfg) loads the individual Data Directory Agent (DDA) modules for posting the following object types: core, inventory, wbem, and patch. Each DDA has its own configuration file (*.dda.cfg) that defines where and how the objects are routed.

See the following topics for more information on how to configure or modify each configuration file.

- [About the Sections in the RMS.CFG File](#) on page 70
- [About the Sections in the CORE.DDA.CFG File](#) on page 72
- [About the Sections in the INVENTORY.DDA.CFG File](#) on page 76
- [About the Sections in the WBEM.DDA.CFG File](#) on page 79
- [About the Sections in the PATCH.DDA.CFG File](#) on page 81
- [Additional Tuning Topics](#) on page 83

 To configure the USAGE.DDA.CFG file, see [Configuring Aggregation](#) in the *CM Application Usage Manager User Guide*.

To edit a Messaging Server or Data Delivery Agent *.cfg file

- 1 Stop the Messaging Server before editing the rms.cfg file. For details, see [Starting and Stopping the Messaging Server](#) on page 48.
- 2 Edit the appropriate *.cfg file using any text editor. By default, the *.cfg files are located at: *Drive:\Program Files\Hewlett-Packard\CM\MessagingServer\etc* for Windows, or */opt/HP/CM/MessagingServer/etc* for UNIX.
- 3 Modify the sections using the information given in the following topics.



All path entries in the configuration files must be specified using forward slashes. This applies to both Windows and UNIX environments.

- 4 Save your changes and restart the Messaging Server.

About the Sections in the RMS.CFG File

The Messaging Server Configuration file, `rms.cfg`, has the following main sections after the header. As of this release, it loads separate modules for data delivery agents (DDAs), whose job is to post the objects to the configured locations.

There are separate data delivery agents for core data (Inventory Manager and Portal objects), wbem data (wbem audit data for Inventory Manager) and patch data (for Patch Manager)

Optional entries in the `rms.cfg` file can include SSL support, a “`msg:register httpd`” section if this Messaging Server is receiving forwarded messages from another Messaging Server, and a “`msg:register default`” section if this Messaging Server is posting messages from a single queue location of `\data\default` (as done in versions prior to v3.0).

Additional Sections in the RMS.CFG File

- **Required packages**

Do not remove the following lines at the top of the `rms.cfg` file

```
package require nvd.msg
package require nvd.httpd
```

- **SSL Configuration Parameters**

If the Messaging Server is SSL enabled, this Overrides Config { } section in the `rms.cfg` is used to define the necessary certificates and parameters for SSL support. It also includes the command `module load tls`, which loads the code necessary to support SSL. For more information, refer to the *HPCA SSL Implementation Guide*.

- **log::init**

Logging configuration. These settings apply to all Messaging Server logging. For details on changing the logging configurations, see [Configuring the Log Level, Log Size and Number](#) on page 85. The log files are located in the Logs directory of where the Messaging Server was installed.

- **loglevel 3**

Default logging level for entries written to the log files. Default is 3. Normally, this is not changed.

- **loglines 200000**

The default number of lines contained in a log before the log is rolled over.

- **loglimit 7**
The default number of rolled logs to keep.
- **Load Data Delivery Agents for posting objects**
Include the following lines at the end of `rms.cfg` to load the data delivery agents needed to post each type of object.
 - **dda.module load core**
Posts CORE message data to a SQL or Oracle database and, optionally, CORE message data to the Portal directory. See [About the Sections in the CORE.DDA.CFG File](#) on page 72 for more information on how to configure the posting of core objects.
 - **dda.module load inventory**
Posts file audit INVENTORY message data to a SQL or Oracle database. See [About the Sections in the INVENTORY.DDA.CFG File](#) on page 76 for more information on how to configure the posting of core objects.
 - **dda.module load wbem**
Posts wbem objects to a SQL or Oracle database. See [About the Sections in the WBEM.DDA.CFG File](#) on page 79 for more information on how to configure the posting of wbem objects.
 - **dda.module load patch**
Post PATCH message data to a SQL or Oracle database. See [About the Sections in the PATCH.DDA.CFG File](#) on page 81 for information on how to configure the posting of patch objects.
- (Optional) **msg::register default, msg::register router, and msg::register <rim|rmp|other>**
These sections, if they exist, define how the Messaging Server handles the messages placed by QMSG in an existing `/data/default` location (or `/data/default queue`). For more information, see [Example 2: Configuring the Messaging Server to Route Objects from a Single \Data\Default Queue](#) on page 108.
 - ▶ These sections are not normally used as of Messaging Server v 3.0. The sections route messages placed into the `\data\default queue` by an earlier version of QMSG. It is still available for customers who are not migrating to the use of multiple queue locations.
- (Optional) Configure the maximum log size and number of logs. Note that these options apply for each Worker assigned to process the `\data\default queue`. See for [Configuring the Log Level, Log Size and Number](#) on page 85 for details.

About the Sections in the CORE.DDA.CFG File

The `core.dda.cfg` file defines where and how to route objects placed in queue locations for core objects. As mentioned previously, the core objects are objects created on the Agent, available during the agent connect process and used in reports. Examples of core objects are ZMASTER, ZCONFIG, and SESSION.

- To activate the `core.dda` module, the command “`dda.module load core`” must be included at the end of the `rms.cfg` file.
- For a Messaging Server co-located with a Configuration Server, the queue locations are folders where the QMSG executable places messages:

Queue folder for core messages: `< CM-CS folder> \data\core`

- For a Messaging Server receiving messages forwarded from another MessagingServer `core.dda` module, URLs define the locations on which to listen for messages:

URL for core messages: `http://localhost:3461/proc/core`

Several configurations are possible.

- Core object data can be routed using ODBC directly to the back-end SQL database. This option is best used when the database is close to the current location.
- Core data can be forward to another Messaging Server. This option is used to place the objects as close as possible to the SQL database, before ODBC posting, in order to avoid slow network response.
- Core messages for the Portal can be routed using HTTP to a Portal Zone, or discarded using the built-in `/dev/null` location.
- Core object data can be routed using HTTP to an existing Inventory Manager Server. This option is no longer supported as of Messaging Server v5, as the Inventory Manager Server has been retired.

The `core.dda.cfg` file has the following main sections after the header and required line:

```
# DO NOT REMOVE FOLLOWING LINE
package require nvd.msg.coreodbc
```

- **msg::register httpd**
This is the HTTPD type for the `core.dda` configuration. If this Messaging Server is receiving messages forwarded from another

Messaging Server, defines the URLHANDLER location on which to look for messages

- **msg::register coreq**

This is the QUEUE type for the `core.dda` configuration. Defines queue used by the how the Messaging Server handles the messages placed by QMSG in the `/data/core` folders.

The parameters are summarized below:

- TYPE of QUEUE defines the Messaging Server location and polling values for picking up messages places in the `/data/<queue>` named by DIR.
- DIR defines the full path of the `/data/core` location. This is set at installation time.
- USE defines where the routing information for each TO label is located.
- POLL and COUNT establish the polling interval and post quantity for the Messaging Server, which determines how often and how many messages are posted at a time. To adjust this, see the topic [Configuring the Poll Interval and Post Quantity](#) on page 83.
- Retry Attempts (DELAY and ATTEMPTS), after maximum retry attempts, a message is automatically discarded

- **msg::register corerouter**

This is the ROUTER type for the `core.dda` configuration. Configures at least one routing assignment for each `-To` type, including:

- a -To CORE.ODBC
- b -To CORE.RIM
- c -To CORE.RMP

The corerouter section enables you to route messages to more than one destination, to another queue type, or to a set disposal location of `/dev/null`.



Default processing of Portal data (as of Messaging Server Version 3.1 and CORE.DDA.CFG Version 3.1) is to re-route the Portal data into its own queue (rmpq) This is discussed below and on page 87.

As of Messaging Server version 3.1, the corerouter section is configured to re-queue the Portal messages into their own queue (rmpq). This permits separate throttling of the CORE messages being posted to the Portal

directory from the CORE messages being posted to an ODBC database. For more information, see [About the Portal Data Queue \(rmpq\) in CORE.DDA.CFG](#) on page 87.



The INVENTORY and WBEM objects are placed in separate queues, and routed according to the `msg::register inventoryrouter` entries in the `inventory.dda.cfg` file and `msg::register wbemrouter` entries in the `wbem.dda.cfg` file, respectively.

The Patch objects are also placed in a separate queue, and delivered according to the `msg::register patchrouter` entries defined in the `patch.dda.cfg` file.

- **msg::register coreodbc**

This is the COREODBC type. Defines a DSN, User, and Password to post core data directly to an ODBC database. For details on the entries, see the topic [ODBC Settings for CORE, INVENTORY and WBEM Objects](#) on page 75.



This section may be configured during the install.

- **msg::register <USE types of HTTP>**

The sections labeled `msg::register rim`, `msg::register rmpqhttp`, as well as `msg::register coreforward` are all examples of HTTP types for the `core.dda` module.

This section defines an external ADDRESS and URL for delivering or forwarding messages using HTTP protocol.

The URL value typically specifies another Messaging Server when using store and forward.



HP recommends using `msg::register COREODBC <USE type of COREODBC>` to post core data directly to the back-end Inventory ODBC database.


The HTTP section is configurable for failover by adding multiple ADDRESS entries, each with a different PRI value. See [Configuring for Failover](#) on page 84 for more information.

- **(Optional) Configure the maximum log size and number of logs.**

Note that these options apply for each Worker assigned to process the specific queue. See [Configuring the Log Level, Log Size and Number](#) on page 85 details.

ODBC Settings for CORE, INVENTORY and WBEM Objects

The following settings are configured in the **msg::register coreodbc** section of `core.dda.cfg`, the **msg::register inventoryodbc** section of `inventory.dda.cfg` and the **msg::register wbemodbc** section of `wbem.dda.cfg`:

DSN	Specify the Data Source Name (DSN) for the Inventory ODBC database. Enclose the entry in quotes.
USER	Specify the user name for the Inventory ODBC database identified in the DSN parameter.
PASS	Specify the password for the user of the Inventory ODBC database. When modifying a password entry, obtain an encrypted entry using the procedure To encrypt a password entry for a database DSN in a configuration file on page 76.
DSN_ATTEMPTS	Number of attempts to connect to the Inventory Manager database DSN. Default is 1.
DSN_DELAY	Delay in seconds between attempts to connect to the Inventory Manager database DSN. Default is 5 seconds.
DSN_PING	Delay in seconds between pings to the database connection to verify the DSN is available. Default is 300 seconds.
AUTOCREATE (wbem.dda.cfg only)	<p>In the WBEMODBC section, a switch to enable the creation of a new SQL file and table in the Inventory ODBC database when a new object class is received. Default is 0.</p> <p>0 – Does not create a SQL file or table entry for a new object class.</p> <p>1 – Creates a new SQL file and table entry for a new object class.</p>
STARTUPLOAD	<p>No longer supported; any coded value is ignored.</p> <p> As of Version 5.0, the SQL tasks to create the tables and load the scripts for the Inventory Database are always performed upon Messaging Server and Data Delivery Agent startup.</p>

To encrypt a password entry for a database DSN in a configuration file

The PASS value in the in all the .cfg files where specification of DSN parameters is necessary has to be encrypted. When the value is entered during the install process, the installation program takes care of encryption. If you need to modify the password, you can use the `nvdkit` utility to create an encrypted password, and specify this encrypted value within the appropriate section of the configuration file. Enclose the encrypted value in quotation marks.

- 1 Open a command prompt and go to the directory where the Messaging Server is installed.
- 2 Enter the following command: `nvdkit`
- 3 At the `%` prompt, type the following command:
`password encrypt <password_value>`
The utility will return an encrypted password value.
- 4 Cut and paste this encrypted password value into the configuration file as the PASS value. Enclose the value in quotation marks.

About the Sections in the INVENTORY.DDA.CFG File

The `inventory.dda.cfg` file defines where and how to route objects placed in queue locations for filepost (file audit inventory) objects.

- To activate the `inventory.dda` module, the command “`dda.module load inventory`” must be included in the `rms.cfg`
- For a Messaging Server co-located with a Configuration Server, the queue location is a folder where the QMSG executable places messages:
Queue folder for inventory messages: `<CM-CS folder>\data\inventory`
- For a Messaging Server receiving messages forwarded from another Messaging Server `inventory.dda` module, URLs define the locations on which to listen for messages:
URL for inventory messages: `http://localhost:3461/proc/inventory`

Several configurations are possible.

- Inventory data can be forward to another Messaging Server. This option is used to place the objects as close to the SQL database as possible before ODBC posting to avoid slow network response.
- Inventory data can be routed using ODBC directly to the back-end Inventory ODBC database. This option is best used when the database is close to the current location.
- In a legacy environment, Inventory data can be routed using HTTP to an existing CM Inventory Manager Server. From there, the Inventory Server can post the messages to the back-end Inventory ODBC database. This option was the previous implementation method, but has been replaced with the direct posting via the data delivery agents into the SQL database.

The `inventory.dda.cfg` file has the following main sections after the header and required line:

```
# DO NOT REMOVE FOLLOWING LINE
package require nvd.msg.inventoryodbc
```

- **msg::register httpd**
This is the HTTPD type for the `inventory.dda` configuration. If this Messaging Server is receiving messages forwarded from another Messaging Server, defines the URLHANDLER location on which to look for messages (separate locations are specified for core and inventory messages).
- **msg::register inventoryq**
This is the QUEUE type for the `inventory.dda` configuration. Defines how the Messaging Server handles the messages placed by QMSG in the `/data/inventory` folders.

The parameters are summarized below:

- TYPE of QUEUE defines The Messaging Server location and polling values for picking up messages places in the `\data\<queue>` named by DIR.
- DIR defines the full path of the `/data/inventory` location. This is set at installation time.
- USE defines where the routing information for each TO label is located.
- POLL and COUNT establish the polling interval and post quantity for the Messaging Server, which determines how often and how many

messages are posted at a time. To adjust this, see the topic [Configuring the Poll Interval and Post Quantity](#) on page 83.

- Retry Attempts (DELAY and ATTEMPTS), after maximum retry attempts, a message is automatically discarded

- **msg::register inventoryrouter**

This is the ROUTER type for the `inventory.dda` configuration. Configures routing assignments for each `-To` type. This section enables you to route messages to more than one destination. It also allows you to route messages to a set disposal location of `/dev/null`. At least one route is specified for each `-To` type:

`-To INVENTORY.ODBC`

- **msg::register inventoryodbc**

This is the COREODBC type for the `inventory.dda` configuration.

Defines an DSN, User, and Password to post Inventory Manager data directly to an ODBC database. For details on the entries, see the topic [ODBC Settings for CORE, INVENTORY and WBEM Objects](#) on page 75.



This section may be configured during the install.

- **msg::register <USE types of HTTP>**

The section labeled `msg::register inventoryforward` is an example of an HTTP section in the `inventory.dda` module.

This section defines an external ADDRESS and URL for delivering or forwarding messages using HTTP protocol.

The URL value specifies another Messaging Server when using store and forward, or the URL for a legacy Inventory Manager Server.



HP recommends using `msg::register inventoryodbc <USE type of inventoryodbc>` to post inventory objects directly to the back-end inventory database. This delivery option has substantial performance benefits over posting the same data to a legacy Inventory Manager server using HTTP, which then posts the data to the back-end database.

The section is configurable for failover by adding multiple ADDRESS entries, each with a different PRI value. See [Configuring for Failover](#) on page 84 for more information.

- (Optional) Configure the maximum log size and number of logs. Note that these options apply for each Worker assigned to process the specific queue. See [Configuring the Log Level, Log Size and Number](#) on page 85 details.

About the Sections in the WBEM.DDA.CFG File

The `wbem.dda.cfg` file defines where and how to route the objects placed in the `\data\wbem` queue location.



The `wbem.dda.cfg` file sections are very similar to the `core.dda.cfg` and `inventory.dda.cfg` file sections.

- To activate the `wbem.dda` module, the command “`dda.module load wbem`” must be included in `rms.cfg`.
- For a Messaging Server co-located with a Configuration Server, the queue location is a folder where the QMSG executable places messages:

Queue folder for `wbem` messages: `<CM-CS folder>\data\wbem`

- For a Messaging Server receiving messages forwarded from another Messaging Server `wbem.dda` module, URLs define the locations on which to listen for messages:

URL for inventory messages: `http://localhost:3461/proc/wbbem`

Several configurations are possible.

- `Wbem` data can be forward to another Messaging Server. This option is used to place the objects as close to the SQL database as possible before ODBC posting to avoid slow network response.
- `Wbem` object data can be routed using ODBC directly to the back-end SQL database. This option is best used when the database is close to the current location.
- `Wbem` data can be routed using HTTP to an existing Radia Inventory Server. From there, the Inventory Server can post the messages to the back-end database. This option is no longer recommended; HP recommends routing data using ODBC directly to the back-end SQL database.

The `wbem.dda.cfg` file has the following main sections after the header and required line:

```
# DO NOT REMOVE FOLLOWING LINE
package require nvd.msg.wbemodbc
```

- **`msg::register wbemhttpd`**
This is the HTTPD type for the `wbem.dda` configuration. If this Messaging Server is receiving messages forwarded from another Messaging Server, defines the URLHANDLER location on which to look

for messages (separate locations are specified for core and inventory messages).

- **msg::register wbemq**

This is the QUEUE type for the `wbem.dda` configuration. Defines how the Messaging Server handles the messages placed by QMSG in the `/data/wbem` location (or `/data/wbem queue`).

The parameters are summarized below:

- TYPE of QUEUE defines The Messaging Server location and polling values for picking up messages.
- DIR defines the full path of the `/data/wbem` location. This is set at installation time.
- USE defines where the routing information for each TO label is located.
- POLL and COUNT establish the polling interval and post quantity for the Messaging Server, which determines how often and how many messages are posted at a time. To adjust this, see the topic [Configuring the Poll Interval and Post Quantity](#) on page 83.
- Retry Attempts (DELAY and ATTEMPTS), after maximum retry attempts, a message is automatically discarded.

- **msg::register wbemrouter**

This is the ROUTER type for the `wbem.dda` configuration. Configures routing assignments for messages with the `-To` type of `wbem.odbc`. This section enables you to route messages to more than one destination. It also allows you to route messages to a set disposal location of `/dev/null`.

- **msg::register wbemodbc**

This is the WBEMODBC type for the `wbem.dda.cfg`. Defines a DSN, User, and Password to post `wbem` inventory data directly to a Inventory Manager ODBC database.



The DSN, User and Password may be configured during the install.

Also defines switches, such as AUTOCREATE, that control when new SQL files and tables are created. For details on the entries, see the topic [ODBC Settings for CORE, INVENTORY and WBEM Objects](#) on page 75.

- **msg::register <USE types of HTTP>**

The section labeled `msg::register wbemforward` is an example of an HTTP section in `wbem.dda.cfg`.

This section defines an external ADDRESS and URL for delivering or forwarding wbem inventory messages using HTTP protocol.

The URL value specifies another Messaging Server when using store and forward, or a legacy Inventory Manager server's URL.



HP recommends using `msg::register WBEMODBC <TYPE of WBEMODBC>` to post wbem and other inventory data directly to the back-end inventory database. This delivery option has substantial performance benefits over posting the same data to the legacy Inventory Manager server, which then posts the data to the back-end database.

This section is configurable for failover by adding multiple ADDRESS entries each with a different PRI value. See [Configuring for Failover](#) on page 84 for more information.

- **(Optional) Configure the maximum log size and number of logs.** Note that these options apply for each Worker assigned to process the specific queue. See [Configuring the Log Level, Log Size and Number](#) on page 85 for more information.

About the Sections in the PATCH.DDA.CFG File

The `patch.dda.cfg` file defines where and how to route the objects placed in the `\data\patch` queue location. Most of the configuration is done automatically during the installation of the Messaging Server.

The `patch.dda.cfg` file has the following main sections after the header and required lines:

```
# DO NOT REMOVE FOLLOWING LINE
package require nvd.msg.patchodbc
package require nvd.msg
```

- **`msg::register patchq`**
This is the QUEUE type for the `patch.dda.cfg`. Defines how the Messaging Server handles the messages placed by QMSG in the `/data/patch` location (or `/data/patch queue`).

The parameters are summarized below:

- TYPE of QUEUE defines a Messaging Server location and polling values for picking up messages.

- DIR defines the full path of the /data/patch message location. This is set at installation time.
 - USE defines where the routing information for the TO PATCH objects are located.
 - POLL and COUNT establish the polling interval and post quantity for the Messaging Server, which determines how often and how many messages are posted at a time. To adjust this, see the topic [Configuring the Poll Interval and Post Quantity](#) on page 83.
 - Retry Attempts (DELAY and ATTEMPTS), after maximum retry attempts, a message is automatically discarded
- **msg::register patchrouter**
This is the ROUTER type for the `patch.dda.cfg`. Configures routing assignments for each `-To` patch type of message. At least one route is specified for each `-To` type:
 - To PATCH
 - To PATCH5
 - **msg::register patchddasummarize**
Translates any ZOBJSTAT patch reporting objects from pre-5.0 agents into the PATCH reporting objects used in Version 5.0 or higher patch reporting. Refer to [Verify the Patch Method Connections and Queue Name](#) for details on the PATCH reporting objects. Not configurable.
 - **msg::register patchddaodbc**
Defines a DSN, User, Password and DBTYPE to post patch data directly to an ODBC database. For details on the entries, see [ODBC Settings for PATCH Objects](#) below.



This section may be pre-configured during the install.

- **(Optional) Configure the maximum log size and number of logs.**
Note that these options apply for each Worker assigned to process the specific queue. See [Configuring the Log Level, Log Size and Number](#) on page 85 for more information.

ODBC Settings for PATCH Objects

The following settings are configured in the **msg::register patchddaodbc** section of `patch.dda.cfg`:

DSN	Specify the Data Source Name (DSN) for the Patch Manager ODBC database. Enclose the entry in quotes.
USER	Specify the user name for the Patch Manager ODBC database identified in the DSN parameter. Enclose the entry in quotes. Default value is {sa}.
PASS	Specify the password for the user of the Patch Manager ODBC database. Enclose the entry in quotes.
DBTYPE	Specify MSSQL for a Microsoft SQL Server, or ORACLE for an Oracle SQL Database. Enclose the entry in quotes. Default is MSSQL.

To encrypt a password entry for a database DSN in a configuration file

The PASS value in the in all the `.cfg` files where specification of DSN parameters is necessary has to be encrypted. When the value is entered during the install process, the installation program takes care of encryption. If you need to modify the password, you can use the `nvdkit` utility to create an encrypted password, and specify this encrypted value within the appropriate section of the configuration file. Enclose the encrypted value in quotation marks.

- 1 Open a command prompt and go to the directory where the Messaging Server is installed.
- 2 Enter the following command: **nvdkit**
- 3 At the `%` prompt, type the following command:
password encrypt <password_value>
The utility will return an encrypted password value.
- 4 Cut and paste this encrypted password value into the `configuration` file as the PASS value. Enclose the value in quotation marks.

Additional Tuning Topics

Configuring the Poll Interval and Post Quantity

By default, the Messaging Server is configured to poll the queue location every 10 seconds and post up to 100 objects at a time. To change the poll

interval, modify the POLL parameter in the appropriate configuration file and msg::register section with a TYPE of QUEUE.

To change the maximum number of objects to be posted at a time, modify the COUNT parameter in the same section.

If the objects being posted are very large, we suggest increasing the POLL interval to give sufficient time to complete the posting.

Configuring for Retry Attempts

The Messaging Server and the Data Delivery Agents are configured to retry any message that fails to post. By default, the Messaging Server will retry posting the message every hour, and make up to 200 attempts. These values are defined by the DELAY and ATTEMPTS entries in the sections of the configuration files that have a TYPE of QUEUE. See [Table 9](#) on page 89 for a list of msg::register sections used to modify QUEUE processing.

▶ After the last attempt, the message is automatically discarded from the queue without being posted.

To calculate the maximum time that a message could stay in the `/data/default` queue, take the DELAY time and multiply it by the ATTEMPTS value. Using the default settings, this is a DELAY of 3600 seconds x 200 ATTEMPTS, or approximately eight days.

You can adjust the DELAY and ATTEMPTS values in configuration files to establish a different maximum time that a message could stay in the `/data/default` queue. Specify the DELAY in seconds.

Configuring for Failover

You can configure the Messaging Server with one or more servers defined for failover support when defining HTTP types. When defining failover servers, each one is assigned a PRI value. If the Messaging Server fails to connect with the first server (that is, the server with the lowest PRI value) it will try the next server on the list (or, the next higher PRI value).

▶ This PRI value is separate from the `-priority` value assigned by QMSG for processing priority. The PRI value and the QMSG `-priority` value are not related.

To set failover in a *.dda.cfg file

Failover support is added by inserting additional ADDRESS entries to the appropriate section of an HTTP type in a DDA cfg file.

The URL entries will be tried in order of PRI (priority) starting with the *lowest* PRI value.

The code sample below shows sample modifications to the msg:register rim section of core.dda.cfg for failover. The code in **bold** was added to define a failover server for Inventory Manager processing.

```
msg::register rim {
    TYPE          HTTP
    ADDRESS       {
        PRI       10
        URL       http://rim1:3466/proc/rim/default
    }
    ADDRESS     {
        PRI      20
        URL      http://rim2:3466/proc/rim/default
    }
}
```

Configuring the Log Level, Log Size and Number

The log files for the Messaging Server (rms.log) reside in the Logs folder of the MessagingServer directory. For example: C:\Program Files\Hewlett-Packard\CM\MessagingServer\logs.

Changing the Logging Level

The log::init section at the beginning of the configuration file establishes the logging level for all Messaging Server logging. The default logging level is 3. Valid levels are 0 (no logging) to 10 (maximum logging level). Normally, the log level is not changed unless requested by a customer support person for troubleshooting purposes. The following lines show the log level increased to 4:

```
log::init {
    -loglevel 4
}
```

Changing the Size and Number of Log Files

The Messaging Server writes entries to a set of log files for each WORKER. There is generally one WORKER attending each queue location. Queue locations include:

```
\data\core
\data\inventory
\data\patch
\data\wbem
```

or

```
\data\default
```

By default, the Messaging Server creates and retains seven log files per worker, each file having a maximum of 5000 lines. The log files are located in the Messaging Server \logs directory.

The following line in the `rms.cfg` file establishes the default logging:

```
-loglines 200000
```

To control the size and number of logs created for each worker, add or modify the following entries below the `log::init` section of the `rms.cfg` file:

```
-loglines maximum_lines
-loglimit maximum number of logs
```

where *maximum_lines* is the maximum number of lines for a given log file. After the maximum is reached, another log file is created, until the *maximum number of logs* specified in the `log.configure -size` entry is reached. After the *maximum number of logs* is reached, the oldest log files are deleted.

The next code sample illustrates an `RMS.CFG` file containing entries to limit each log file to 1000 lines, and allow up to 10 log files to be retained.

```
log::init {
    -loglevel 3
    -loglines 1000
    -loglimit 10
}
```

Configuring the Messaging Server to Discard or Drain Messages

The location of `/dev/null` is built into the Messaging Server for discarding messages. When the `USE` parameter is set to `/dev/null` in any of the `ROUTER` type sections of the Messaging Server or Data Delivery Agent

configuration files, the messages being processed will be successfully discarded without an error.

Example: Discarding messages for the Portal

For example, to discard all RMP messages placed in the `\data\core` queue (these messages have a TO label of `CORE.RMP`), specify the following **ROUTE** in the `msg::register corerouter` section of `core.dda.cfg`:

```
msg::register corerouter {
    TYPE    ROUTER
    . . .
    ROUTE   {
            TO      CORE.RMP
            USE     /dev/null
    }
}
```

Example: Draining a Message Queue

As another example, to quickly drain an entire queue, temporarily replace `USE router` in the `msg::register` section for the queue with `USE /dev/null`. See [Table 9](#) on page 89 for a list of the configuration files and sections that control each queue type. After draining the queue, reset it back to `USE router`.

Configuring the Messaging Server to Route Portal Messages

About the Portal Data Queue (`rmpq`) in `CORE.DDA.CFG`

The default `core.dda.cfg` configuration re-queues CORE messages that are to be posted to the Portal directory into its own queue, named `rmpq`. This allows for separate throttling of the CORE messages being posted via HTTP to the Portal directory, as opposed to the CORE messages being posted using ODBC to another database.

The following code shows the sections from `core.dda.cfg` used for this purpose.

```
# Requeue and process just RMP data to throttle the data flow

msg::register rmpq {

TYPE            QUEUE

DIR            ../ConfigurationServer/data/rmp
USE            rmpqrouter
```

```

POLL          10
COUNT       30
DELAY        3600
ATTEMPTS     200
}

msg::register rmpqrouter {
TYPE          ROUTER

ROUTE        {
TO           CORE.RMP
USE          rmpqhttp
}
}

msg::register rmpqhttp {
TYPE          HTTP

ADDRESS      {
PRI          10
URL          http://localhost:3471/proc/xml/obj
}
}

```

Restoring Routing for Portal Messages

If you initially installed the Messaging Server to discard Portal messages, use the steps below to begin routing Portal data to a Portal Server and Port.

To modify `core.dda.cfg` for posting Portal data

- 1 Use a text editor to edit the `core.dda.cfg` file, located in the `etc` folder of the Messaging Server directory.
- 2 Look for the section starting with `msg::register corerouter`, and then find the entry for the `ROUTE` defining `CORE.RMP` messages. RMP data that is being discarded will show the following entry with a `USE` value set to `/dev/null`:

```

ROUTE        {
TO           CORE.RMP
USE          /dev/null

```

- 3 Change the `USE` value from `/dev/null` to `rmpq`, as shown below:

```

TO           CORE.RMP
USE        rmpq

```


- Next, locate the `msg::register rmpqhttp` section near the end of the `core.dda.cfg` file, and find the default URL entry, shown below:

```
msg::register rmpqhttp {
    TYPE          HTTP

    ADDRESS      {
        PRI       10
        URL       http://localhost:3466/proc/xml/obj
    }
}
```

- Edit the URL value of `localhost:3466` to indicate the host and port of your Portal server. For example, a Portal with a hostname of `PORTALSVR` on port 3471 is defined with the following URL entry:

```
URL      http://PORTALSVR:3471/proc/xml/obj
```

Host names can be specified using an IP address or a DNS name.

- Save your changes and restart the Messaging Server. For details, see [Starting and Stopping the Messaging Server](#) on page 48.

Disabling Processing of Messages in a Queue

The objects in a disabled queue are not polled or posted. You may want to disable processing during peak agent connect periods if resources are in contention, or if you know a target server is down.

You can re-enable the processing at night or during slower periods to allow the Messaging Server to transfer the messages.

To disable queue processing using `WORKERS -1` (minus 1) value

To disable a queue from being polled and its contents posted, set a `WORKERS` value of -1 (minus one) at the end of the appropriate `msg::register` section for that queue.

Table 9 Configuration File and Section Used to Modify Queue Processing

Queue	Configuration File in MessagingServer\etc folder	msg::register section
\data\core	core.dda.cfg	msg::register coreq
\data\inventory	core.dda.cfg	msg::register inventoryq
\data\patch	patch.dda.cfg	msg::register patchq

Queue	Configuration File in MessagingServer\etc folder	msg::register section
\data\wbem	wbem.dda.cfg	msg::register wbemq
\data\default (in earlier Versions)	rms.cfg	msg::register default

To add a WORKERS value to create multiple processes (WORKERS)

- 1 Use a text editor to open the appropriate *.cfg file for the Messaging Server or Data Delivery Agent processing the queue to be disabled. Table 9 on page 89 identifies the configuration file to use for each queue type.
- 2 Locate the 'msg::register' section named in Table 9; it will be defined with { TYPE QUEUE }.
- 3 Add or modify a line for WORKERS with a value of -1 (minus 1) to the end of the section. A sample entry for disabling a wbem queue is shown below with a WORKERS value of -1.

```
msg::register wbemq  {
    TYPE           QUEUE

    DIR
    C:/Progra~/Hewlet~/CM/ConfigurationServer/data/wbemq
    USE           router

    POLL          10
    COUNT         100
    DELAY         3600
    ATTEMPTS      200
    WORKERS       -1
}
```

- 4 Save your changes and exit the editor.

To enable a disabled queue

To enable a previously disabled queue, change the WORKERS value in the msg::register section of appropriate configuration file from -1 to 1. The number of WORKERS indicates the number of independent and lightweight processes to be started for this queue. The default configuration uses 1, which is the HP-recommended value for a Messaging Server that is processing multiple queues with Data Delivery Agents.

Modifying the Priority in which Messages are Processed



With the adoption of specific queues for each object type, the priority feature is no longer applicable. However, the code for processing messages in increasing priority order has not been disabled.

To modify the priority in which messages are processed, see the earlier topic [How Priority Establishes Messaging Server Processing Order](#) on page on page 62.

4 Troubleshooting

At the end of this chapter, you will:

- Understand how to resolve common error messages in the `rms.log` files. This log file is located in the `Logs` directory of the root `MessagingServer` installation directory.
- Understand how to resolve failed posts.

Troubleshooting the Messaging Server



If your environment uses Core and Satellite servers, first read the Core and Satellite Servers Getting Started Guide as the installation, configuration, and troubleshooting information in that guide may override the information in this guide.

The Messaging Server log file is located in the Logs directory of the root MessagingServer installation directory.

Problem: Log indicates no route defined or failed delivery

Your Messaging Server log includes WARNING and ERROR messages indicating 'no route defined for default' and 'failed to deliver to default', as shown below.

```
Warning: router1: To: default, From: <CM-CS>@<CM-CS_hostname>, subject: - no route  
defined for default
```

```
Error: MSG/QUEUE: q2: To: default, From: <CM-CS>@<CM-CS_hostname>, subject: - failed to  
deliver to default
```

Solution:

These messages indicate that one or more QMSG calls in ZTASKEND are missing a -to parameter and or value. When this happens, the word 'default' becomes the -to value for that message. Since the Messaging Server does not have a route defined for messages labeled with a -to value of default, it cannot route the message and writes these warning and error messages to the log.

The solution is to review the QMSG calls in ZTASKEND and add any missing -to parameters or missing values. As delivered, QMSG -to values include: -to core.rim, -to core.rmp, -to inventory, and -to inventory.wbem, although there may be others. For details, see the [QMSG Method Syntax](#) on page 60.

Problem: Error 404 or 500

You receive Error 404 (page not found) or error 500 (server internal error) for all attempted posts to a RIM Server.

Solution:

Review the *.sql files located in the /etc/sql folder of your Integration Server. Check the bottom of your sql files to see if there is a commented out section that looks like:

```
#sql::url . . .
```

The solution for Error 404 or Error 500 is to remove the # (pound sign) to un-comment this line.

- If you do not have any customizations, you can move all of the *.sql files out of the /etc/sql directory (place them in an outside location), and then stop and start the Integration Server. This unpacks the new .sql files, which should fix the error.
- If you have more than one customization, use the following steps to correct the problem and still keep your customizations:
 - a Locate any *.sql files that you have customized in the /etc/sql folder.



The HP-delivered default .sql files will be located in the /etc/sql/hp directory. The data delivery agents will load the customized files from the /etc/sql directory first, if a file is not found in the /etc/sql directory, it loads it from the /etc/sql/hp directory. Therefore the customized files will always take precedence over the default files.

- b Delete the remaining *.sql files from the /etc/sql/hp folder.
- c Restart the CM Integration Server to unpack a new set of *.sql files into the default location of /etc/sql/hp.
- d Go to where you placed the customized *.sql files, and un-comment the sql::url line at the bottom of each file.
- e Restart the HPCA Integration Server service.

If you have any questions regarding this document or this code, please refer to the [HP Software Support](#) web site.

Summary

- Review the common error messages and solutions given in this topic to troubleshoot Messaging Server problems.
- Failed posts can be the result of the line "sql::init" being commented out in one or more files in the `/etc/sql` folder of your RMI server's installation directory.

A Optional Messaging Server Configurations

The Messaging Server can be adapted to meet various messaging needs. While this appendix is not comprehensive, it presents a few simple models for using the Messaging Server in alternative configurations. These configurations include:

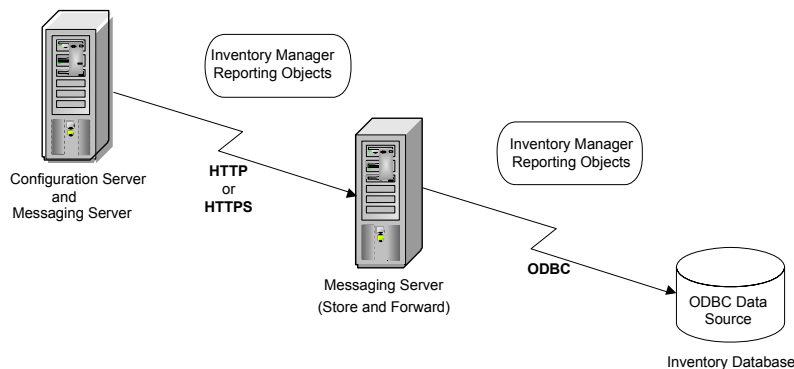
- Example 1: Configuring the Messaging Server for Store and Forward on page 98.
- Example 2: Configuring the Messaging Server to Route Objects from a `Single\data/default Queue`.
- Example 3: Configuring Messaging Server to Route to Multiple Queues on page 111.

▶ Example 3 is for illustrative purposes only. We advise you to contact [HP Software Technical Support](#) to discuss your individual needs before making substantial changes to your Messaging Server configuration. In addition, remember to fully test any new configurations in a non-production environment, including the use of stress-tests that duplicate production volumes.

Example 1: Configuring the Messaging Server for Store and Forward

The Messaging Server includes store and forward capabilities that allow you to create a multiple-Messaging Server environment. When a Messaging Server is used for store-and-forward processing, messages are forwarded from one Messaging Server to another, before being sent to their final destination. This is illustrated in [Figure 4](#) below.

Figure 4 Store and Forward "Hop"



The concept of Store and Forward is moving the messages through the network to a location closer to where the work will be done on them. The messages in essence "hop" from Messaging Server to Messaging Server. The forwarding Messaging Servers route data using HTTP or HTTPS.

The first data queue drains very quickly, since there is no data "processing" taking place on the sending or receiving end. For example, you can configure the Messaging Server on the Configuration Server to forward all inventory messages to another, remote, Messaging Server. This configuration drains the inventory messages from the Configuration Server location quickly, freeing up Configuration Server resources for agent-resolution tasks.

The following topics explain how to create a store and forward messaging environment:

- Installing and Configuring a "Receiving" begins on page 99.
- Configuring a Messaging Server to Forward Messages begins on page 102.

Installing and Configuring a "Receiving" Messaging Server

The concept of Store and Forward is moving the messages through the network to a location closer to where the work will be done on them. The messages in essence "hop" from Messaging Server to Messaging Server. The downstream Messaging Server can actually reside on the same server as the SQL server avoiding any problems using ODBC across the network.

Using the Store and Forward, messages will be forwarded to a "Receiving" Messaging Server from a "Sending" Messaging Server using HTTP or HTTPS. To accomplish this, `rms.cfg` and the various `*.dda.cfg` files need to be configured for either sending or receiving messages after installation.

When using DDAs, the sender and receiver modifications need to be made to each of the `dda.cfg` modules that are processing messages.



Tip! Diagram your network topology noting the IP address and port configurations of the initial receiver (this will be the server where the Configuration Server Database is hosted), of any intermediate Messaging servers, and of the final destination Messaging Server being used to post data. For the final destination server, also note how you are posting the data: via ODBC to a SQL-compliant server or via HTTP to the Portal.

About the Messaging Server Receiver

The Messaging Server is configured for receiving messages in its { TYPE HTTPD } section. The default `rms.cfg` file includes the following settings in that section:

```
msg::register httpd {  
  
    TYPE          HTTPD  
  
    PORT          3461  
    USE           default  
  
    URL           /proc/rim/default  
    URL           /proc/xml/obj  
}
```

In these `rms.cfg` settings:

- PORT 3461 is the default listening port for receiving incoming messages.
- USE specifies the name of the QUEUE type to deposit messages into when messages are received.

- URL specifies which URL strings will be accepted.

Using the default settings above and assuming the server name and port this listener resides on is TESTSERVER and 3461, a message sent with the following URL will be received and placed in the queue defined by default:

```
http://TESTSERVER:3461/proc/rim/default
```

Configuring the Receivers for the .dda Modules

Little, if any, reconfiguration is needed to setup a receiving Messaging Server. There is only one listening port specified for a Messaging Server instance. The Port specification is made in `rms.cfg`.

The HTTPD section type within the `rms.cfg` and each `dda` module is the pre-configured listener section that accepts and queues messages received from another Messaging Server. It normally needs no reconfiguration.

This is an example of the default listener section for the CORE object messages in `core.dda.cfg`:

```
msg::register corehttpd {
    TYPE          HTTPD
    URLHANDLER    {
        USE        coreq
        URL        /proc/core
    }
}
```

If we assume the Messaging Server name is TESTSERVER and the listening Port is 3461, then the previous DDA configuration allows CORE messages sent with the following URL to be placed in the queue identified as `coreq`:

```
http://TESTSERVER:3461/proc/core
```

Below is an example of the default listener section for the PATCH object messages in `patch.dda.cfg`.

```
msg::register patchhttpd {
    TYPE          HTTPD
    URLHANDLER    {
        USE        patchq
        URL        /proc/patch
    }
}
```

If we assume the Messaging Server name is TESTSERVER and the listening Port is 3461, then the previous DDA configuration allows patch messages sent with the following URLs to be placed in the queue identified as patchq:

```
http://TESTSERVER:3461/proc/patchq
```

Note: If necessary, you can specify more than one pair of queue and URL entries for a DDA section. To do this, you can code an additional URLHANDLER.

The messages received on the previously named URLs are placed in the directory defined by the DIR parameter of the QUEUE section:

For example, if the coreq section is defined as:

```
msg::register coreq {  
  
    TYPE          QUEUE  
    DIR           {../ConfigurationServer/data/core}  
    USE           corerouter  
    POLL         10  
    COUNT        100  
    DELAY        3600  
    ATTEMPTS     200  
  
}
```

then the messages received on the url:

```
http://TESTSERVER:3461/proc/core
```

will be queued in the directory ../ConfigurationServer/data/core, which is the directory specified by coreq.

Running the Installation for a Receiving Server and DDAs

- Run the install, and load any Data Delivery Agents that will be handling messages on the receiving server. For example, if this server is only being used to receive and post Patch objects to the Patch database, (and is not processing any Inventory objects), you can just select the Patch DDA during the install.
- When prompted for the Message Directory to Scan, choose a location on the existing server that includes the same `\data\<queue_name>` directory convention as the sending Messaging Server. If the directory you specify does not exist, it will be created.

For example, when prompted for the Core Message Directory to Scan, you could type:

C:\MessagingServer\data\core

And when prompted for the Patch Message Directory to Scan, you could type:

C:\MessagingServer\data\patch

The scan directory entries define the DIR value in the TYPE QUEUE section of the `rms.cfg` and `dda.cfg` files.



All directory entries in `*.cfg` files require forward slashes (/).

```
msg::register coreq {  
    TYPE          QUEUE  
  
    DIR           {C:/MessagingServer/data/core}  
    USE          corerouter
```

- Once started, the receiving Messaging Server queues messages sent to it in the newly created "Message Directory to Scan" location. The Messaging Server and DDAs will scan and process these messages as specified by the appropriate data delivery agent configuration file.
- If posting Patch data to a database on Oracle, ensure the `patch.dda.cfg` file specifies DBTYPE "ORACLE". Refer to page 43 for more information.
- Review the Data Delivery Agent configuration file(s) for the appropriate routing of the messages being received. See the topics in the chapter [Configuring and Tuning the Messaging Server](#) on page 53.

Configuring a Messaging Server to Forward Messages

To Forward messages, you generally configure the Messaging Server or DDAs to empty their queues using HTTP, and send the messages to a receiver Messaging Server.

Let us assume the Messaging Server was installed with DDAs for each of the Inventory Manager objects (CORE, INVENTORY, and WBEM DDAs), and we want to forward those messages to another Messaging Server installed with those DDAs listening downstream. We want the DDAs on the downstream Messaging Server to post the data to a SQL-compliant database using ODBC.

By default, the `dda.cfg` files are configured to route the messages identified with the "TO" destination label of CORE.ODBC to the section defined with TYPE COREODBC.



The new ZTASKEND release specifies the `-to` parameter as `CORE.ODBC`, check the ZTASKEND specification to make sure of this.

To have the DDAs configured to forward messages to a DDA on another Messaging Server, you must tell the DDA to route them using HTTP to another DDA on a downstream server. The needed changes are given in the following procedures.

Forwarding CORE, INVENTORY and WBEM Messages

To configure a Messaging Server to forward CORE, INVENTORY and WBEM messages

Use this procedure to modify the configuration files on an existing Messaging Server so CORE, INVENTORY and WBEM messages are forwarded to another Messaging Server—instead of to their final destination.

The procedures to forward PATCH messages are similar. They begin on page 106.

- If you are using data delivery agents, make these changes to the configuration file for *each* agent currently processing the messages that are to be forwarded.
- If you are not using data delivery agents, make these changes to the `rms.cfg` file.
- These steps can be adjusted to forward all or some of the message types: CORE, INVENTORY and WBEM.

- 1 Stop the Messaging Server service (`rms`).
- 2 Edit the appropriate `cfg` file for the Server or Agent that is currently processing those messages that are to be forwarded. For example, to forward the `core.odbc` messages, edit the `core.dda.cfg` file. For more information on each type of `*.cfg` file, see [Editing the Configuration File](#) on page 63.
- 3 Locate the TYPE ROUTER section in the `*.cfg` file. For example: `'msg::register corerouter'` is the TYPE ROUTER section in the `core.dda.cfg` file. For a data type that is being routed to a section with TYPE `*ODBC`, switch the USE parameter to the label for a `{ TYPE HTTP }` section.

Table 10 on page 105 shows the USE parameter being switched from coreodbc to coreforward. This reroutes the CORE.ODBC data from a { TYPE COREODBC } section to a {TYPE HTTP } section.



The core, inventory and wbem *.dda.cfg files include TYPE HTTP sections named coreforward, inventoryforward, and wbemforward, respectively.

The coreforward section is defined to route the CORE.ODBC data, using HTTP, to the Messaging Server and Port named in the URL. The next step is to replace the default values in the URL to specify the downstream Messaging Server (see the next step).

- 4 In the coreforward section, specify the IP address or host name in the URL entry to identify the receiving Messaging Server and port number. The default Store and Forward *port* number for the Messaging Server is 3461.

Table 10 on page 105 shows the changes made to the coreforward and corerouter sections to forward the CORE.ODBC data to a downstream server named DOWNSTREAM.

- a In the ROUTER section labeled corerouter, the route for CORE.ODBC data was changed from 'USE coreodbc' to 'USE coreforward'.
- b In the HTTP section labeled coreforward, the URL specifying the ADDRESS was modified to include the host name of our Messaging Server:

```
URL http://DOWNSTREAM:3461/proc/core
```

- 5 Comment out the entire section for TYPE COREODBC or WBEMODBC.

- a In core.dda.cfg, comment out the `msg::register coreodbc {}` section.
- b In inventory.dda.cfg, comment out `msg::register wbemodbc {}`.
- c In wbem.dda.cfg, comment out `msg::register wbemodbc {}`.

Table 10 on page 105 shows the `msg:register coreodbc` section commented out.

Table 10 Sample forwarding modifications to the CORE.DDA.CFG file

Original CORE.DDA.CFG Entries	Edits to Forward CORE.ODBC Data
<pre> .. msg::register corerouter { TYPE ROUTER ROUTE { TO CORE.ODBC USE coreodbc } ROUTE { TO CORE.RIM USE corerim } ROUTE { TO CORE.RMP USE /dev/null } } msg::register coreodbc { TYPE COREODBC DSN "" USER "" PASS "{DES}:0" DSN_ATTEMPTS 1 DSN_DELAY 5 DSN_PING 300 } </pre>	<pre> .. msg::register corerouter { TYPE ROUTER ROUTE { TO CORE.ODBC USE coreforward } ROUTE { TO CORE.RIM USE corerim } ROUTE { TO CORE.RMP USE /dev/null } } #msg::register coreodbc { # TYPE COREODBC # DSN "" # USER "" # PASS "{DES}:0" # DSN_ATTEMPTS 1 # DSN_DELAY 5 # DSN_PING 300 #} msg::register coreforward { TYPE HTTP ADDRESS { PRI 10 URL http://DOWNSTREAM:3461/proc/core } } </pre>

- 6 To forward message types that are currently being routed to a TYPE HTTP section, such as 'msg::register rim' and 'msg::register rmp', all you need to do is modify the URL specified in those sections. Change the host and port in the URL entry to specify the host and port of the receiving Messaging Server. Keep the rest of the URL entry the same.

The format for various URL entries is given below:

For msg::register rim and msg::register corerim:
URL

http://<MsgSvr_hostname:port>/proc/rim/default

For msg::register rmp and msg::register corermp:

URL http://<MsgSvr_hostname:port>/proc/xml/obj

Where:

MsgSvr_hostname can be specified using an IP address or a DNS name, and

Port is the store and forward port for the Messaging Server, normally 3461.

- 7 Save the changes to the configuration files that were edited. On the downstream server hosting the Messaging Server, the listener on port 3461 requires an HTTPD section in its configuration file to accept the URLs and place the accepted message in the specified queue. For more information on the HTTPD section, see [About the Messaging Server Receiver](#) on page 99.
- 8 Restart the Messaging Server (rms) Service. For details, see [Starting and Stopping the Messaging Server](#) on page 48.

Forwarding PATCH Messages

To configure a Messaging Server to forward PATCH messages to another Messaging Server

Perform these procedures to modify the configuration of a Messaging Server installed with the Patch DDA to now forward patch messages using HTTP to a receiving Messaging Server. This manual configuration takes a few minutes.

Have the listening, or receiving, Messaging Server already installed, as discussed on page 99. You will need to supply the receiving server's hostname or IP address and listening port number (default is 3461) during the steps to configure the forwarding messaging server, below.

- 1 Stop the Messaging Server service (rms) that is being reconfigured to forward patch messages.
- 2 Edit the `patch.dda.cfg` file on the Messaging Server that is currently processing the messages to be forwarded.
- 3 Locate the TYPE ROUTER section named 'msg::register patchrouter' and change the following two USE parameters to specify `patchforward`:
 - a Change `USE patchddasummarize` to `USE patchforward`
 - b Change `USE patchddaodbc` to `USE patchforward`
- 4 Go to the end of the `patch.dda.cfg` file, and locate the section: 'msg::register patchforward'.
- 5 In the patchforward section, change the host and port in the URL entry to specify the host and port of the Messaging Server to receive the patch messages. Keep the rest of the URL entry the same.

The format for the Patch URL entry is:

```
URL      http://<MsgSvr_hostname:port>/proc/patch
```

Where:

MsgSvr_hostname can be specified using an IP address or a DNS name, and

Port is the store and forward port for the Messaging Server, normally 3461.

- 6 Save the changes and restart the Messaging Server (rms) service.

This completes the steps to configure a Store and Forward Messaging Server for patch data.

Example 2: Configuring the Messaging Server to Route Objects from a Single \Data\Default Queue

Prior versions of the Messaging Server (versions below 3.x) routed objects placed by QMSG into a single queue location of `\data\default`. The topics that follow discuss the sections in the `rms.cfg` file that are needed to post messages from the `\data\default` queue to the appropriate locations.



The `msg::register default` section is not normally used in this version of the Messaging Server. However, it is still supported for customers who are not migrating to the use of multiple queue locations and data directory objects.

- **msg::register default (optional)**
Defines how the Messaging Server handles the messages placed by QMSG in the `/data/default` folder (or, the `/data/default` queue).
- The parameters are defined in About the Sections in the CORE.DDA.CFG File, and summarized below:
 - DIR defines the full path of the `/data/default` location. This is set at installation time.
 - USE defines where the routing information for each TO label is located.
 - POLL and COUNT establish the polling interval and post quantity for the Messaging Server, which determines how often and how many messages are posted at a time. To adjust this, see the topic [Configuring the Poll Interval and Post Quantity](#) on page 83.
 - Retry Attempts (DELAY and ATTEMPTS), after maximum retry attempts, a message is automatically discarded
 - WORKERS parameter (optional). If not specified, a default of one WORKER is used. You can add the following line to increase the number of WORKERS:

```
WORKERS 2
```


Set to 2, or up to 4. You can disable processing by setting WORKERS to -1.
- **msg::register router**
Configures routing assignments for each -To type. This section enables you to route messages to more than one destination. It also allows you to

route messages to a set disposal location of `/dev/null`. At least one route is specified for each `-To` type:

`-To inventory`

`-To inventory.wbem`

`-To rim.core`

`-To rmp.core`

`-To patch` (Patch Manager prior to Version 5.00)

`-To patch5` (Patch Manager Version 5.00 and above)



Match the routing for each `-to` type of object being posted by the QMSG executable. These can be verified by browsing the `ZTASKEND` rexx file for calls to QMGS.

- **msg::register <USE type>** (for example: `msg::register rim`, `msg::register rmp`, etc.) These are the initial definition of HTTP locations established during installation.
 - Configure the section for Failover. See [Configuring for Failover](#) on page 84.
 - Configure the section to Discard Messages. See [Configuring the Messaging Server to Discard or Drain Messages](#) on page 86.
- (Optional) Configure the maximum log size and number of logs. Note that these options apply for each Worker assigned to process the `/data/default` queue. See [Configuring the Log Level, Log Size and Number](#) on page 85 for details.

Configuring the Register Default Section

Use the following table to modify the parameters in the `msg::register` default section of `rms.cfg`.

Table 11 RMS.CFG Parameters used to Define the /Data/Default Queue

Parameter	Default	Definition
TYPE	QUEUE	Registration type. <i>Do not change this value.</i>
DIR	<code>../ConfigurationServer/data/default</code>	Directory where your Configuration Server (through <code>QMSG.exe</code>) will queue XML objects to post. Edit the DIR value to reflect the full path of your <code>data/default</code> directory <i>using forward slashes</i> for both Windows and UNIX platforms.
USE	router	Internal setting telling the program what process to use. <i>Do not change this setting.</i>
POLL	10	Delay in seconds for polling the local store of objects to be posted. Increase this value to support the posting of large objects, such as those for Operational reports.
COUNT	100	The number of XML objects that will be posted at each POLL interval.
DELAY	3600	Amount of time in seconds to retry a failure.
ATTEMPTS	200	How many times the Messaging Server will try to post a message before discarding it. Note: <code>DELAY * ATTEMPTS</code> gives the maximum time a message will stay in the queue before automatic discard. Using the default values of DELAY and ATTEMPTS, a message is discarded after approximately 8 days of failed posting attempts.
WORKERS	1	Optional entry. Number of asynchronous, lightweight processes to create for this queue. To drain a queue more quickly, we recommend using WORKERS set to 2. A second worker doubles the processing power of a single Messaging Server configuration, with each worker performing a separate POLL and

Parameter	Default	Definition
		<p>COUNT.</p> <p>Using more than 4 WORKERS is NOT recommended.</p> <p>Note: Set to -1 (minus 1) to temporarily disable a queue from being processed.</p>

Example 3: Configuring Messaging Server to Route to Multiple Queues

This example configures the Messaging Server to route messages from a single queue to multiple queues based on their destination. A message's destination is defined by the **-to** parameter value passed from QMSG and stored in the meta-data in the message file.

In the configuration shown in [Figure 5](#) on page 112, all messages will be placed in the standard location (the directory `C:/Program Files/Hewlett-Packard/CM/ConfigurationServer/data/default`) when they are created by QMSG. We want to have the Messaging Server route these messages into separate message queues before they are processed. In the code sample that follows, we define the routes for `CORE.RIM` and `CORE.RMP` messages to use `queue1` and `queue2` definitions. The new `queue1` and `queue2` definitions direct the `CORE.RIM` messages to the `C:/rim` directory and the `CORE.RMP` messages to the `C:/rmp` directory.

Additional sections must be coded in `rms.cfg` to complete the routing of the messages. However, this example illustrates the basic concept of routing messages from a single queue to multiple queues, before routing them to processing destinations.

Additional sections must be coded in `rms.cfg` to complete the routing of the messages. However, this example shows the basics of how you can route messages from a single queue to multiple queues, before routing to processing destinations.

The following is a sample `rms.cfg` configuration sorting messages into multiple queues.

Figure 5 Sample RMS.CFG configuration sorting messages into multiple queues

```
msg::register default {
    TYPE      QUEUE

    DIR      ../ConfigurationServer/data/default
    USE      router1


    POLL     10
    COUNT    100
    DELAY    3600
    ATTEMPTS 200
}
msg::register router1 {
    TYPE      ROUTER

    ROUTE    {
        TO    CORE.RIM
        USE    queue1
    }

    ROUTE    {
        TO    CORE.RMP
        USE    queue2
    }
}
msg::register queue1 {
    TYPE      QUEUE

    DIR      C:/rim
}
msg::register queue2 {
    TYPE      QUEUE


    DIR      C:/rmp
}
```



Example 4: Configuring Data Delivery Agents to Route to Multiple DSNs using ODBC

This example configures the CORE Data Delivery Agent to post CORE messages to two different DSNs using ODBC. This is done by creating two separate queues and posting the data from each queue to a separate DSN.

In the example that follows, following modifications are made:

- 1 The `corerouter` section is modified to route each CORE.ODBC message to two DSNs via two separate queues. The first ROUTE is defined to use `coreodbcq1` and the second ROUTE is defined to use `coreodbcq2`.
- 2 Processing of the first queue continues as follows:
 - a A QUEUE for `coreodbcq1` is defined and routes its messages to `coreodbcq1router`.
 - b A ROUTER for `coreodbcq1router` is defined. It routes the messages to a COREODBC section named `coreodbc1`.
 - c A COREODBC section is defined named `coreodbc1`. This is where the messages are posted to the first DSN using ODBC.
 To encrypt the DSN password entry, see [To encrypt a password entry for a database DSN in a configuration file](#) on page 76.
- 3 Processing of the second queue basically duplicates the first:
 - a A QUEUE for `coreodbcq2` is defined and routes its messages to `coreodbcq2router`.
 - b A ROUTER named `coreodbc2router` is defined. It routes messages to a COREODBC section named `coreodbc2`.
 - c A COREODBC section is defined with the name `coreodbc2`. Here is where the second DSN is defined and the messages are posted using ODBC.

See [Figure 6](#) which follows for a sample `core.dda.cfg` file configured with these sections.

Figure 6 Sample CORE.DDA.CFG configured to post data to multiple DSNs

#select core.dda.cfg sections modified to route each message to 2 DSNs via 2 separate queues

```
msg::register corerouter {
    TYPE          ROUTER

    ROUTE        {
        TO        CORE.ODBC
        USE        coreodbcq1
    }

    ROUTE        {
        TO        CORE.ODBC
        USE        coreodbcq2
    }

    ROUTE        {
        TO        CORE.RMP
        USE        rmpq
    }
}

#first queue - coreodbcq1 - routes to first DSN

msg::register coreodbcq1 {
    TYPE          QUEUE

    DIR          {../ConfigurationServer/data/coreodbcq1}
    USE          coreodbcq1router

    POLL         10
    COUNT        100
    DELAY        3600
    ATTEMPTS     200
}
```

```

}

msg::register coreodbcq1router {
    TYPE            ROUTER

    ROUTE          {
        TO          CORE.ODBC
        USE         coreodbc1
    }
}

msg::register coreodbc1 {
    TYPE            COREODBC

    DSN             "RIMSQL"
    USER            "sa"
    PASS            "{DES}:0"
    DSN_ATTEMPTS   1
    DSN_DELAY       5
    DSN_PING        300
}

#second queue - coreodbcq2 - routes to second DSN

msg::register coreodbcq2 {
    TYPE            QUEUE

    DIR             {../ConfigurationServer/data/coreodbcq2}
    USE             coreodbcq2router

    POLL           10
    COUNT          100
    DELAY          3600
    ATTEMPTS       200
}

```

```

msg::register coreodbcq2router {
    TYPE          ROUTER

    ROUTE        {
        TO        CORE.ODBC
        USE        coreodbc2
    }
}

# added additional COREODBC type for second DSN
msg::register coreodbc2 {
    TYPE          COREODBC

    DSN           "RIMSQL2"
    USER          "sa"
    PASS          "{DES}:0"
    DSN_ATTEMPTS  1
    DSN_DELAY     5
    DSN_PING      300
}

```

This is the end of the topics for alternative configurations.

Index

A

- agent object processing, 54
- AGENT_SELFMAINTENANCE phase, 57
- AGENT_REPORTING, 57
- AGENT_REPORTING phase, 57
- Application Management Profiles, 29
- ATTEMPTS value, 65, 84
- AUTOCREATE
 - in WBEMODBC section, 75

B

- BOOTSTRAP phase, 57
- BuildAlways, 60

C

- calls to QMSG, 56
- CATALOG_RESOLUTION phase, 57
- CATALOG_RESO, 57
- CLISTAT, 17
- CLISTATS object, 59
- CM Application Usage Manager
 - configuring for aggregation, 47
- configuration files, location, 63
- copyright notices, 2
- Core Data Delivery Agent
 - configuration window, 31
 - configuring, 72
 - configuring during install, 34
 - CORE data routing options, 72
 - ODBC Settings, 75
 - routing RMP data, 88
 - routing RMP messages, 88
- CORE message data, 17

- CORE object data, 18
- CORE.DDA.CFG. *See* Core Data Delivery Agent
- CORE.ODBC message, 113
- coreforward, 20
- coreforward section, 104
- COREODBC section type, 67
- corerouter section, 104, 113
- critical objects, 57
- customer support, 6

D

- Data Delivery Agents, 16
 - configuring, 63
 - during install, 25
 - on a receiving server, 100
 - to forward messages, 103
 - defined, 17
 - installing, 25
 - installing with Messaging Server, 29
 - usage, 47
- data queue, 87
- DBTYPE
 - for Patch ODBC on Oracle, 43
- Default Message Directory, 30
- DELAY time, 65, 84
- disabled queue, enabling, 90
- disabling message processing, 89
- discarding messages, 86
- draining messages, 86
- draining the message queue, 87

E

- Error 404 or 500, 94

ERROR message, failed to deliver to default, 94

F

failover, 84

 configuring with HTTP routing, 84

FILEPOST object, 18, 59

filepost.tcl, 18

FILTER section type, 67

forwarding messages, 102

H

HPCA Core, 12

HPCA Patch Manager, post-installation procedures,
44

HPCA Satellite, 12

HTTP section type, 66

HTTPD section type, 66

HTTPS

 configuring for, 46

 HTTPS_PORT in URL, 46

HTTPS section type, 66

I

installation

 message directories to scan, 30

 select Data Delivery Agents, 29

 Store & Forward Port, 32

 task overview, 26

 verifying, 50

 with Store and Forward, 101

Inventory Data Delivery Agent

 configuring, 76

 configuring during install, 36

 ODBC Settings, 75

 routing options, 77

Inventory Manager

 critical core objects, 57

INVENTORY message data, 17

INVENTORY.DDA.CFG. *See* Inventory Data
Delivery Agent

inventoryforward, 20

InventoryQueue, 60

J

JOBPARM object, 59

JOBSTAT object, 59

JOBTASK object, 59

L

legal notices

 copyright, 2

 restricted rights, 2

 warranty, 2

log files

 log level, changing, 85

 size and number, changing, 86

M

Messaging Server

 configuring, 63

 Data Delivery Agents, introduction, 17

 installing, 24

 Inventory data, routing options, 18

 processing, 15

 processing on the Configuration Server, 13

 store and forward configurations, 98

 store and forward, introduction, 19

 tuning topics, 83

Messaging Service

 as a Windows service, 17

 starting and stopping, 48

meta data files, 16

msg::register, 110

multiple DSNs, sample configuration, 113

N

nvdkit, encrypt password, 76, 83

O

ODBC section type, 67

ODBC Settings
 configuring for ODBC routing, 75
 for Patch objects, 82

Oracle
 DBTYPE for Patch, 43
 setting DBTYPE in Patch.DDA.CFG, 43

P

password encryption, 76, 83

Patch Data Delivery Agent
 configuring, 81
 configuring during install, 38

PATCH message data, 17

PATCH.DDA.CFG. *See* Patch Data Delivery Agent

PATCH_BUSTATUS
 modifying, 44

PATCH_DEERROR
 modifying, 44

PATCH_DESTATUS
 modifying, 44

PATCH_PASTATUS
 modifying, 44

PATCH_RESTATUS
 modifying, 44

PATCHDDAODBC section type, 68

PATCHDDASUMMARIZE section type, 68

Poll interval and post quantity, 83

Portal
 critical core objects, 57
 discarding messages for, 87
 routing messages to, 88

post-installation procedures, 39

PRI value, 84

Q

QMSG, 16, 94
 called
 from PATCH methods, 55
 from ZTASKEND, 55

 destinations and Data Delivery Agent locations,
 61
 how it formats messages, 54
 message syntax, 60
 priority parameter, 62

QMSG call syntax, 56

queue
 disabling, 89
 draining, 87
 names on Configuration Server, 16

QUEUE section type, 65

R

restricted rights legend, 2

retry attempts, configuring, 84

rmpq, 87

rmpqhttp, 87

rmprouter, 87

RMS.CFG, 26
 configuring, 70
 dda module load statements, 71
 logging configurations, 70
 modifying, 69
 required packages, 70
 routing sections, 71
 SSL Configuration Parameters section, 70

rms.log, 85
 PID for HPCA Messaging Service, 49

ROUTER section type, 65

S

section types, 64

SERVICE RESOLUTION phase, 57

SQL database, 17

STARTUPLOAD not supported, 19, 75
 in COREODBC section, 67

Store and Forward, 19, 98
 about the receiving server, 99
 about the sending server, 102
 configuring a forwarding server and DDAs, 103

- installing a receiving server and DDAs, 101
- port number, 32
- sample CORE.DDA.CFG to forward messages, 105

support, 6

T

taskend.tcl, 18

technical support, 6

Thin clients

- Windows CE core.dda requirement, 30

tuning, 83

U

URLHANDLER, 73, 77, 79, 101

Usage Data Delivery Agent

- configuring, 47
- installing, 29

USE parameter, 86, 104

V

Vulnerability Management

- core.dda requirement, 30

W

WARNING, no route defined for default, 94

warranty, 2

Wbem Data Delivery Agent, 37

- configuring, 79
- configuring during install, 37

- ODBC Settings, 75
- routing options, 79

WBEM message data, 17

WBEM.DDA.CFG. *See* Wbem Data Delivery Agent

WBEMAUDT object, 59

wbemforward, 20

WBEMODBC section type, 68

Windows CE thin clients

- core.dda requirement, 30

Windows service, 17

WORKERS, how to disable, 89

X

XML files, 16

Z

ZERROR, 57

ZERRORM, 57

ZMTHPRMS

- queue value, 32

ZMTHPRMS, modifying queue name for patch, 44

ZOBJSTAT, 17

ZTASKEND, 15, 94

- about, 55

- critical core object processing, 57

- modifying always objects, 59

- modifying for critical objects, 59

- RMS 2.x and 3.x processing, 60