HP Configuration Management

Messaging Server

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

Software Version: 5.1x

Installation and Configuration Guide

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

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

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 for release 5.10 and 5.11.

Table 2 indicates changes made to this document for release 5.00.

Table 3 indicates changes documented with previous editions.

Table 1   Document Changes for Release 5.10 and 5.11

Chapter	Version	Changes
Chapter 2	5.11	Page 31, Table 5, AIX was added to the supported Operating Systems with this important note: "Note: AIX-support does not allow for direct ODBC posting to a database; it requires messages to be forwarded to a non-AIX Messaging Server for posting."
Chapter 2	5.11 Mar 08	Page 31, Messaging Server Installation Tips, added tip to review Release Notes prior to running the installation.
Chapter 2	5.10	Page 36, Select the Data Delivery Agents to Install, select the new usage.dda during the installation for Windows to support aggregation of CM Application Usage Manager data files.
Chapter 2	5.10	Page 59, Verify the Patch Method Connections and Queue Name, Patch Manager 5.10 includes an additional reporting object: PATCH_PASTATUS.

Chapter	Version	Changes
Chapter 2	5.10	Page 63, Configure the USAGE.DDA to Aggregate and Forward Usage Data, following installation, refer to the Configuring Aggregation topic in the *CM Application Usage Manager Guide* for procedures on how to configure the usage.dda to support aggregation.

Appendix A	5.10 and 5.00	Page 119, Forwarding CORE, INVENTORY and WBEM Messages, added the following configuration requirement to Table 13:
		
		Comment out the entire DDA section with a TYPE COREODBC. For example, in the core.dda.cfg, comment out the #msg::register coreodbc { } section as shown below:
		
		``
		#Section commented out for a forwarding configuration.
		#msg::register coreodbc {
		# TYPE COREODBC
		
		# DSN ""
		# USER ""
		# PASS "{DES}:0"
		# DSN_ATTEMPTS 1
		# DSN_DELAY 5
		# DSN_PING 300
		#}
		``

Table 2   Document Changes for Release 5.00

Chapter	Version	Changes
Chapter 1	5.00	Page 24, Creating SQL Tables for the Inventory Database, all SQL tables for the Inventory Database are created when the Messaging Server is started and the SQL commands are loaded when the Data Delivery Agents are started. **Note:** 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.

Chapter	Version	Changes
Chapter 2	5.00	Page 30, Platform Support, supported platforms have changed. Installation on a Windows platform requires Windows 2000 Server or Windows Server 2000. UNIX supported platforms have changed.
Chapter 2	5.00	Page 32, Installation Procedures for Windows and UNIX: Type `./setup` to launch a local installation on a UNIX machine.
Chapter 2	5.00	Page 42, Configure the CM Messaging Server Store and Forward Port. Many installation panels previously used to route messages from the data\default queue have been eliminated. Route messages using the data delivery agents, instead. To receive messages forwarded from another Messaging Server, enter a Store and Forward port number.
Chapter 2	5.00	Page 55, Configure ODBC Drivers for HP-UX, Linux, or Solaris, new post-installation procedure details how to configure the provided Datadirect Connect ODBC drivers with HP-UX, Linux, or Solaris platforms.
Chapter 2	5.00	Page 59, Set the DBTYPE for a Patch ODBC Database on Oracle, added this post-installation task to post messages to a Patch ODBC databases running on Oracle. You must switch the DBYPE value in the patch.dda.cfg file from "MSSQL" to "ORACLE".
Chapter 2	5.00	Page 59, Verify the Patch Method Connections and Queue Name, Patch Manager now reports data using several objects (DEERROR, BUSTATUS, DESTATUS and RESTATUS), instead of the single object ZOBJSTAT.
Chapter 3	5.00	Page 83, Table 11, STARTUPLOAD configurable parameter is no longer supported by the following section types: COREODBC and WBEMODBC; any entry in a DDA configuration file will be ignored.
		Page 91, ODBC Settings for CORE, INVENTORY and WBEM Objects, the STARTUPLOAD configuration parameter is no longer supported. All SQL tasks to create the CM Inventory Database tables and load the SQL commands are performed when the Messaging Server and Data Delivery Agents are started.

Chapter	Version	Changes
Chapter 3	5.00	Page 97, About the Sections in the PATCH.DDA.CFG File, the `patchodbc` and `patchfilter` sections have been replaced by `patchddasummarize` and `patchddaodbc` sections.
Chapter 3	5.00	Page 99, ODBC Settings for PATCH Objects, new DBTYPE parameter must specify the database type: "MSSQL" for Microsoft SQL Server or "ORACLE" for an Oracle SQL database.
Appendix A	5.0	Page 114, Example 1: Configuring the CM Messaging Server for Store and Forward, topic revised to include Patch Store and Forward considerations.
		Page 115, About the CM Messaging Server Receiver, modified to include patch.dda.cfg examples. No special configuration is required.
		Page 122, Forwarding PATCH Messages, new topic explains how to reconfigure the patch.dda.cfg file to forward patch messages.

Table 3   Document Changes from Earlier Releases

Chapter	Version	Changes
Chapter 1	3.0	Page 18, Features and Capabilities, added the ability to "Maintain multiple data queues on Store and Forward Messaging Servers."
Chapter 1	3.0	Page 18, Benefits over Previous Implementations, new topic highlights the new benefits of using multiple queues, Data Delivery Agents, and the new routing options of ODBC and HTTPS.

Chapter	Version	Changes
Chapter 1	3.1	Page 18, Benefits over Previous Implementations: Version 3.1 adds new benefits, including: • Configurable control over when SQL tables are created and commands are generated—either upon *first message delivery* (HP recommended and default) or at Messaging Server startup. • A new queue to control and throttle the posting of only Management Portal data from the Core Data Delivery Agent. • Support for customized routing of messages to multiple DSNs using ODBC.
Chapter 1	3.0	Page 22, About the Data Delivery Agents, new topic explains the role of the new Data Delivery Agents in routing the objects from the Core, Wbem, and Patch data queues to the appropriate locations.
Chapter 1	3.0	Page 23, About Routing Inventory Data, new topic discusses the choices available for routing inventory objects—either directly to an Inventory database using ODBC, or indirectly to a RIM server using HTTP.
Chapter 1	3.1	Page 23, About the SQL Database Tables and Scripts, new topic explains the new file locations for HP-delivered files, where to maintain your custom files and scripts, and how to efficiently control when SQL tasks are performed.
Chapter 1	3.0	Page 24, About Using Store and Forward, new topic and Figure summarizes typical ways to use a Store and Forward Messaging Server to locate the data objects close to an ODBC database before posting it.
Chapter 2	3.0	Page 31, Tips: new topic.
Chapter 2	3.0	Page 32, Installation Procedures for Windows and UNIX: the installation has been completely rewritten to accommodate the installation of Data Delivery Agents. Use the Overview of Installation Tasks on page 32 to guide you through the install.

Chapter	Version	Changes
Chapter 2	3.1	Page 36, Table 7, new table summarizes which DDAs to install by product.
Chapter 2	3.0	Page 55, Post-Installation Procedures: new topics. Use these procedures to verify the correction configurations for Patch, as well as enable HTTP routing using SSL.
Chapter 2	3.1	Page 63, Revert to a CM Messaging Server Configuration for Single-queue Processing, new topic identifies tasks and topics needed to support the Messaging Server use of a single \data\default queue and non-ODBC processing, as was done for Messaging Server version 2.x.
Chapter 2	3.1	Page 66, Verify Installation: expanded topic discusses how to use the rms.log to verify the Messaging Server and Data Delivery Agents are running, and the nvdkit processes are available for each queue worker.
Chapter 3	3.0	Page 71, About the Patch Method for Collection, new topic to explain the method used to call QMSG for Patch data.
Chapter 3	3.0	Page 71, About the ZTASKEND REXX method: modified topic to reflect the ZTASKEND v1.9 changes, and the routing of objects to separate data queues on the Configuration Server.
Chapter 3	3.0	Page 79, Configuring the CM Messaging Server: contents are changed substantially due to the loading of data delivery agent modules. Very few items are configured directly in the rms.cfg file as of version 3.0.
Chapter 3	3.0	Page 81, Table 11 gives a glossary of all configurable section TYPES and their parameters.
Chapter 3	3.1	Page 83, Table 11, new STARTUPLOAD configurable parameter added to the following section types: COREODBC and WBEMODBC. STARTUPLOAD controls whether SQL tasks are performed upon the first message delivery (default) or upon Messaging Server startup. Page 83, Table 11, a new AUTOCREATE configurable parameter was added for the WBEMODBC section type.

Chapter	Version	Changes
Chapter 3	3.0	Page 88, About the Sections in the CORE.DDA.CFG File: new topic explains how to configure the Data Delivery Agent to route core objects to an Inventory Database or Server, as well as the Management Portal.
Chapter 3	3.1	Page 86, Additional Sections in the RMS.CFG File, entries for **log.configure -limit** and **log.configure -lines** were added to this topic.
Chapter 3	3.1	Page 89, About the Sections in the CORE.DDA.CFG File, Version 3.1 modifies how Management Portal data is routed from the mgs::register corerouter section into a new queue, named rmpq.
Chapter 3	3.1	Page 91, ODBC Settings for CORE, INVENTORY and WBEM Objects, new STARTUPLOAD and AUTOCREATE configuration parameters were added to this topic.
Chapter 3	3.0	Page 92, About the Sections in the INVENTORY.DDA.CFG File: new topic explains how to configure the Data Delivery Agent to route filepost objects to an Inventory Database or Server.
Chapter 3	3.0	Page 95, About the Sections in the WBEM.DDA.CFG File: new topic explains how to configure the Data Delivery Agent to route wbem objects to an Inventory Database or Server.
Chapter 3	3.0	Page 97, About the Sections in the PATCH.DDA.CFG File: new topic explains how to configure the Data Delivery Agent to route patch objects to a Patch database.
Chapter 3	3.0	Page 100, Additional Tuning Topics: most tuning topics were modified to address tuning the parameters in the appropriate data delivery agent configuration file, as well as the rms.cfg file.
Chapter 3	3.1	Page 104, About the CM Portal Data Queue (rmpq) in CORE.DDA.CFG, new topic shows the new sections in CORE.DDA.CFG used to re-queue only Management Portal messages before they are posted using HTTP.

Chapter	Version	Changes
Appendix A	3.0	Page 114, Example 1: Configuring the CM Messaging Server for Store and Forward: this revised topic explains how to modify both the Messaging Server and Data Delivery Agent configuration files for store and forward configurations.
Appendix A	3.0	Page 114, Example 2: Configuring the CM Messaging Server to Route Objects from a Single \Data\Default Queue, new topic explains how to use sections in the rms.cfg file to route data objects placed in a single data\default queue, when Data Directory Agents are *not* used. Note: This topic was relocated to the Appendix due to the use of multiple data queues as of Messaging Server v 3.0.
Appendix A	3.0	The earlier configuration example: *Configuring the Messaging Server for Custom Named Queues* has been deleted. The adoption of multiple Data Delivery Agents with individual queue names has eliminated the need for this customization.
Appendix A	3.0	Page 129, Example 4: Configuring Data Delivery Agents to Route to Multiple DSNs using ODBC, new example illustrates how to customize a core.dda.cfg file to create two queues to route messages to two separate DSNs. This example duplicates message delivery of CORE.ODBC messages to two different target databases.

The chapter *Migrating Inventory Processing to Use QMSG and the Messaging Server* has been deleted from this guide. For migration information, refer to the *CM Messaging Server Migration Guide*. This guide is located in the \migrate folder of where the Messaging Server is located on the CM Infrastructure media.

Support

You can visit the HP Software support web site at:

www.hp.com/managementsoftware/services

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

At the end of this chapter, you will:

- Know the benefits of the HP Configuration Management Messaging Server (CM Messaging Server).

- Understand CM Messaging Server processing using message queues and data directory objects.

- Become familiar with the CM Messaging Server Data Delivery Agent modules.

About the CM Messaging Server

The CM Messaging Server is a robust messaging service that provides a means to forward data from one piece of the CM 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 CM servers by type. Its job is to monitor predefined data queues and dynamically route data objects to one or more external destinations. The CM Messaging Server provides retry, rerouting, and failover capabilities to ensure all data is transferred efficiently and reliably.

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

Features and Capabilities

The CM Messaging Server provides an efficient and flexible messaging system that can be used by CM 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 CM Messaging Server to another one (store and forward capability).

- Maintain multiple data queues on Store and Forward CM 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 CM Messaging Server to post object data directly to a CM Inventory Manager database via ODBC eliminates the need for the previous CM Inventory Manager Server.

- Eliminates bottlenecks on the Configuration Server.

- Reliability of processing CM Inventory and CM 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 CM 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 CM Portal data.

- New support for customized message routing to multiple DSNs using ODBC.

CM Messaging Server Processing on the CM Configuration Server

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

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

Figure 1 on page 20 provides an overview of CM Messaging Server Processing.

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

The CM Messaging Server refers to CORE objects as the standard CM agent objects that are exchanged between the agent and the CM 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, and RESTATUS agent objects collected for CM Patch Manager processing.

Figure 1   CM Messaging Server Processing
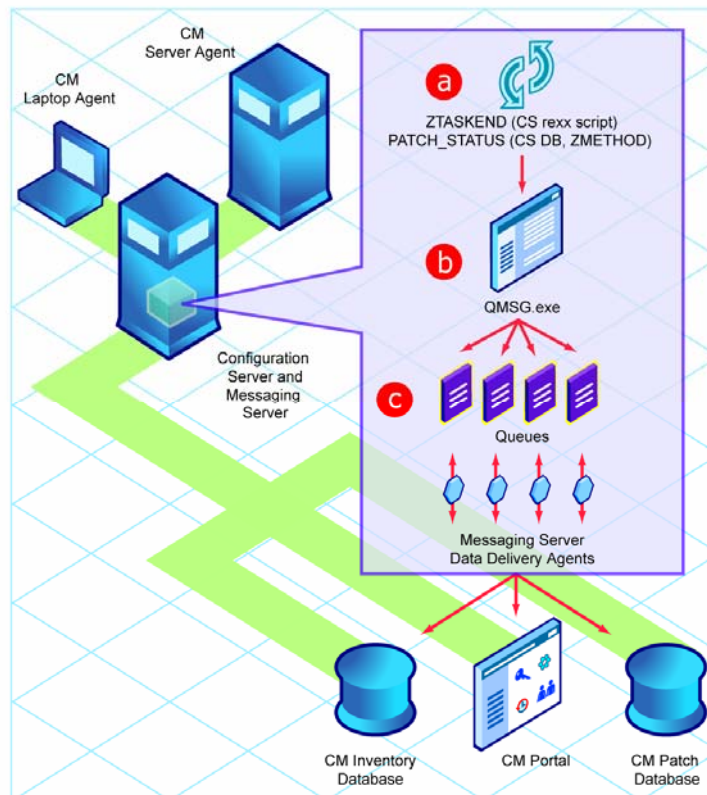
Legend

a   ZTASKEND is called

b   QMSG.EXE assembles object data

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

a ZTASKEND is Called
On the CM 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 (PATCH_DEERROR, PATCH_BUSTATUS, PATCH_DESTATUS, and PATCH_RESTATUS) in the PRIMARY.SYSTEM.ZMETHOD class of the CM Configuration Server Database calls 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 CM 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 CM 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 CM Messaging Server is configured to use individual Data Delivery Agents (DDAs) to process messages from these queues. Table 4 below lists the Data Delivery Agents that operate on each data queue location.

c CM Messaging Server Data Delivery Agents poll the queues and transfer data
The CM 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 4   Data Queues and Data Delivery Agents

Data Queue	Data Delivery Agent
`..\data\core`	`core.dda`

Data Queue	Data Delivery Agent
`..\data\inventory`	`inventory.dda`
`..\data\wbem`	`wbem.dda`
`..\data\patch`	`patch.dda`
`..\data\default` *(prior to MsgSvr v3.0)*	*none – processed by base MsgSvr*

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

- 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 24.

The CM 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 a CM Inventory Manager database, as well as CORE object data to a CM 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 CM 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 CM Messaging Server supports direct ODBC posting of CM 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 24.

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 CM Inventory Manager Server (which is now retired). The default definitions for these tables and associated SQL queries are found in the following CM 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 CM 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 CM Inventory Database and load the SQL commands into memory are performed when the CM 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 CM Messaging Server includes store and forward capabilities that allow you to forward messages to another CM Messaging Server using HTTP or HTTPS. This is the AIX-supported solution. In addition, if the Messaging Server is not located close 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 CM Messaging Server supports forwarding and receiving messaging using multiple queues.

Figure 2 on page 25 illustrates a typical configuration that forwards messages to another CM Messaging Server, prior to posting data to the Inventory database using ODBC.

Figure 2   Store and Forward CM Messaging Server
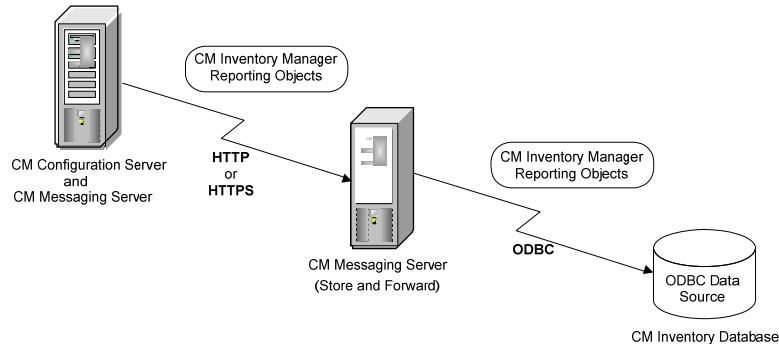
1   The CM Messaging Server on the CM Configuration Server is configured to have the core, inventory, and wbem data delivery agents forward the data to another CM Messaging Server using HTTP or HTTPS. This configuration makes use of the *coreforward, inventoryforward,* and *wbemforward* sections that are provided in the data delivery agent configuration files.

2   The Store and Forward CM Messaging Server is located close to the CM 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 CM Messaging Server post the data objects to the CM Inventory Manager database using ODBC.

For more information on this topic, see

About this Guide

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

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

- **Configuring and Tuning the CM Messaging Server**
  This chapter describes how the CM Configuration Server ZTASKEND

rexx and the QMSG executable work hand-in-hand with the CM Messaging Server. It also discusses how to configure the CM 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 CM Inventory, CM Portal, and CM 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 CM 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 CM Messaging Server routes the object data collected from CM agents and placed into queues into the appropriate CM server or SQL database. Messages can also be forwarded to another CM 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 CM Messaging Server

At the end of this chapter, you will:

- Know how to install the HP Configuration Management Messaging Server (CM Messaging Server).
- Be able to verify the installation of the CM Messaging Server.

CM Messaging Server Installation

Before you install the CM Messaging Server, identify the server where the CM Messaging Server will reside. Among the available choices are the same physical server that is running the Directory Services (or SQL database), the CM 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 CM Messaging Server solution. When making the choice of servers for installation of the CM Messaging Server, please bear in mind the recommended best practice of locating the CM 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 CM Messaging Servers in a Store and Forward configuration. Configuring the CM Messaging Server for Store and Forward is discussed in Appendix A, Optional CM Messaging Server Configurations. The Store and Forward capability requires that the CM 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 CM Messaging Server. Install the CM Messaging Server from the Extended Infrastructure directory on the CM 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 100.

Platform Support

For detailed information about supported platforms, refer to the release note document that accompanies this release.

The CM Messaging Server runs on the Windows and UNIX platforms listed in Table 5 on page 31. These include all platforms on which the CM Configuration Server runs.

Table 5   Supported operating systems and minimum levels

OS Name	OS Version and Chipset Architecture
AIX	5.1, 5.2, 5.3 **Note**: AIX-support does not allow for direct ODBC posting to a database; it requires messages to be forwarded to a non-AIX Messaging Server for posting.
Windows	2000 Server, Service Pack 3 on x86
	Server 2003, Service Pack 1 on x86
	Server 2003 x64 on AMD64/EM64T
HP-UX	Version 11.0, 11.11 on PA-RISC-2
	Version 11.23 or 11.31 on PA RISC-2 and Intel Itanium
Red Hat Enterprise Linux ES, AS	Version 3.0, 4.0, or 5.0 on x86 and AMD64/EM64T
SuSE Linux Enterprise Server	Version 9 and 10 on x86 and AMD64/EM64T
Sun Solaris	Version 8, 9, and 10 on SPARC

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 CM Messaging Server or each of the four 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 CM Messaging Server install using the same installation program. Once a DDA is installed and its data-specific directory exists, the ZTASKEND method on the CM Configuration Server automatically redirects the messages to the new directory location.

Installation Procedures for Windows and UNIX

> ⚠️ If you have previously installed the CM 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`.

Overview of Installation Tasks

Because the CM Messaging Server installation supports prior and current configuration options, there can be many prompts for information during the

install that follows. Use Table 6 below as a roadmap to the sequence and contents of the prompts.

Table 6   CM Messaging Server Installation -- Task Overview

Task	Page	Notes and Tips
1 Launch install, accept license and select the CM Messaging Server directory	34	Rename an existing `rms.cfg` file before you begin.
2 Select Data Delivery Agents to Install	35	Each DDA selected here (except for the usage.dda) adds the following windows: • 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 usage.dda, refer to the *CM Application Usage Manager User Guide*.
3 Identify Message Directories to Scan (Message Folder Locations)	37	You are always prompted for the CM 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 `ZMTHPRMS -queue` value in five `PATCH methods`. See page 60 for details.
4 Configure the CM Messaging Server Store and Forward Port	42	Identify a store and forward port the CM Messaging Server can receive messages on. Default is 3461.
5 Configure the Core Data Delivery Agent (core.dda.cfg)	43	Displays if the CORE DDA was selected in Step 2. Also routes CM Portal objects, if desired.
6 Configure the Inventory Data Delivery Agent (inventory.dda.cfg)	46	Displays if INVENTORY DDA was selected in Step 2. Routes the FILEPOST objects to a CM Inventory Manager database.

Task	Page	Notes and Tips
7 Configure the Wbem Data Delivery Agent (wbem.dda.cfg)	49	Displays if WBEM DDA was selected in Step 2.
8 Configure the Patch Data Delivery Agent (patch.dda.cfg)	51	Displays if PATCH DDA was selected in Step 2.
9 Review Installation Summary and Finish	54	Allows review before proceeding with the install.
10 Perform post-installation tasks	55	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 CM Messaging Server directory

1 From the CM 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 CM 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.
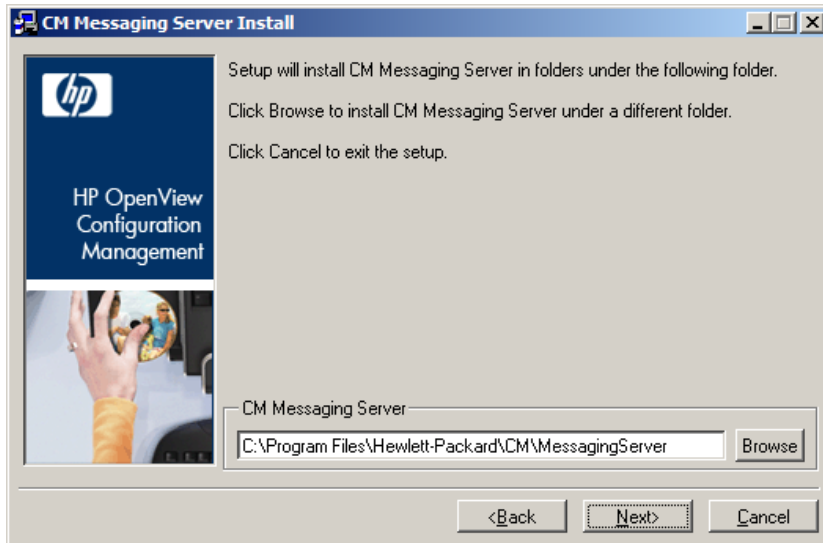
6   Use this window to select the folder where you want to install the CM Messaging Server.

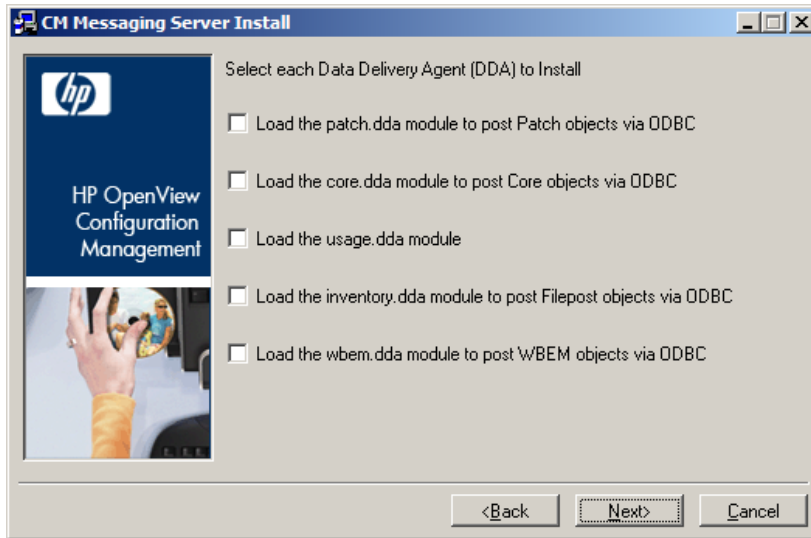— Click **Next** to accept the default installation folder.

or

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

7   Click **Next**.

Task 2   Select Data Delivery Agents to Install

The 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 CM Messaging Server. See Table 7 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 *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.

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

Table 7   DDAs to install by Configuration Management Product

Product	Data Delivery Agents to Install
Application Management Profiles	`core.dda`
CM Application Usage Manager	`usage.dda` (Windows only)
CM Inventory Manager	`core.dda, inventory.dda` and `wbem.dda`
CM Portal	`core.dda`
CM Patch Manager	`patch.dda`

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

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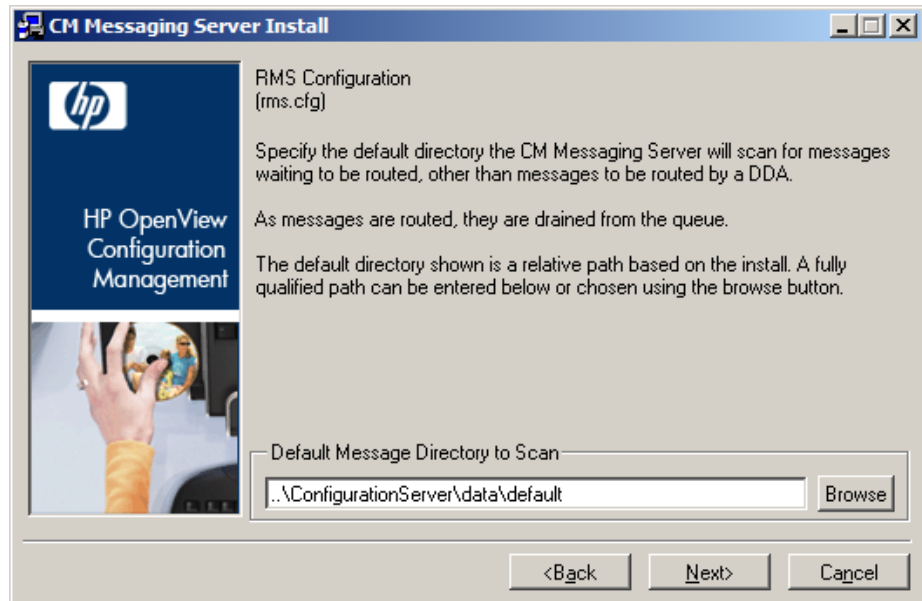
9   Accept the default or click **Browse** to select the directories where the CM Messaging Server should scan for any messages that are *not* being routed by a Data Delivery Agent. Even if all Data Delivery Agents are installed, this Default Message Directory must exist.

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

This directory is created upon start-up of the CM Messaging Server if the directory doesn't exist.
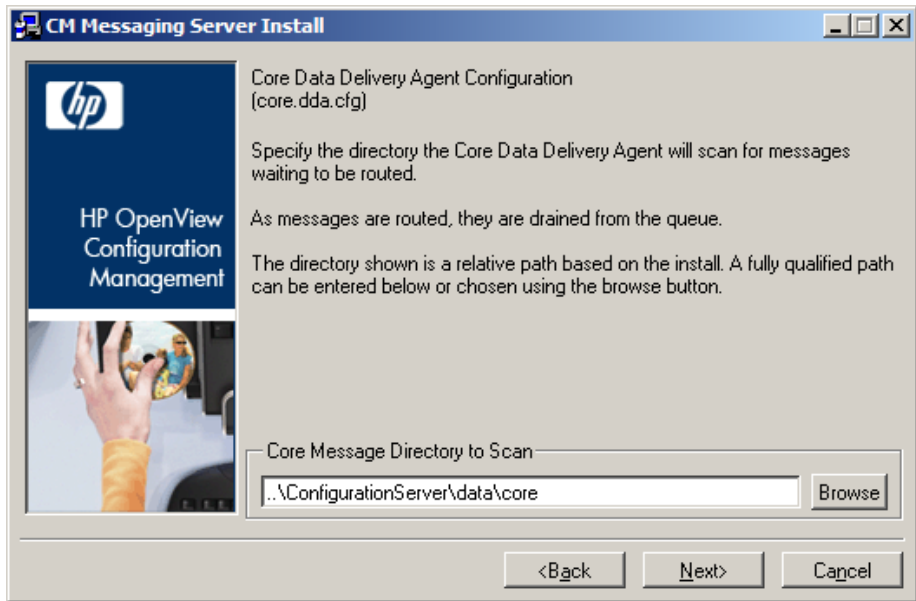
> If necessary, adjust the directory path to your CM Configuration Server, but keep the \data\default folder names.

10 Click **Next**.

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

If the Core Data Delivery Agent was selected, the Core Data Delivery Agent Configuration window opens.

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 CM Configuration Server is installed. This directory will be created if it doesn't already exist when the CM Messaging Server starts. Changes to this directory name have to be coordinated with changes to ZTASKEND REXX for a CM Messaging Server co-resident with the CM Configuration Server.

> If necessary, adjust the directory path to your CM 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.

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

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

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

14 Click **Next**.

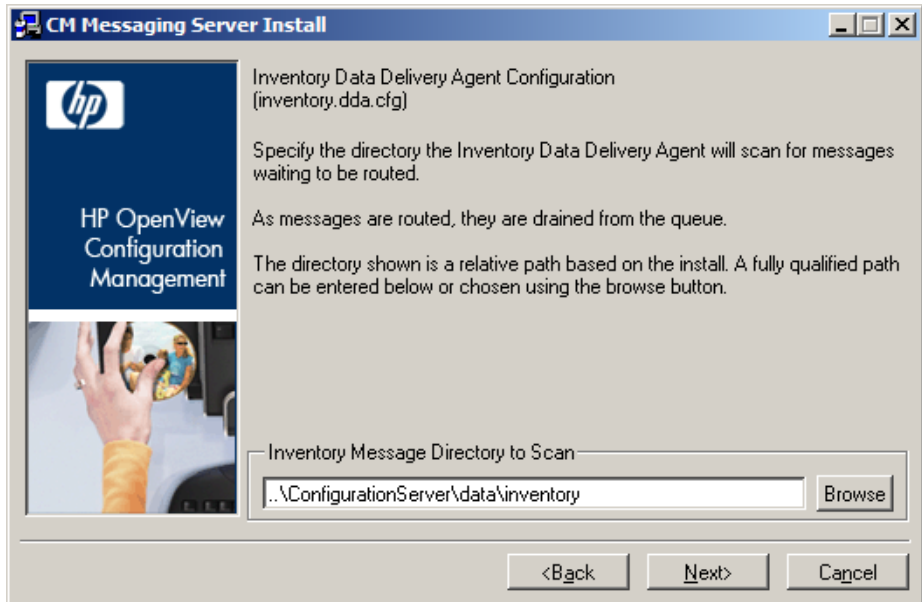If the Wbem Data Delivery Agent was selected, the Wbem Delivery Agent Configuration window opens.

15 Accept the default location, or click **Browse** to select the directory where the Wbem Data Delivery Agent should scan for any wbem messages.
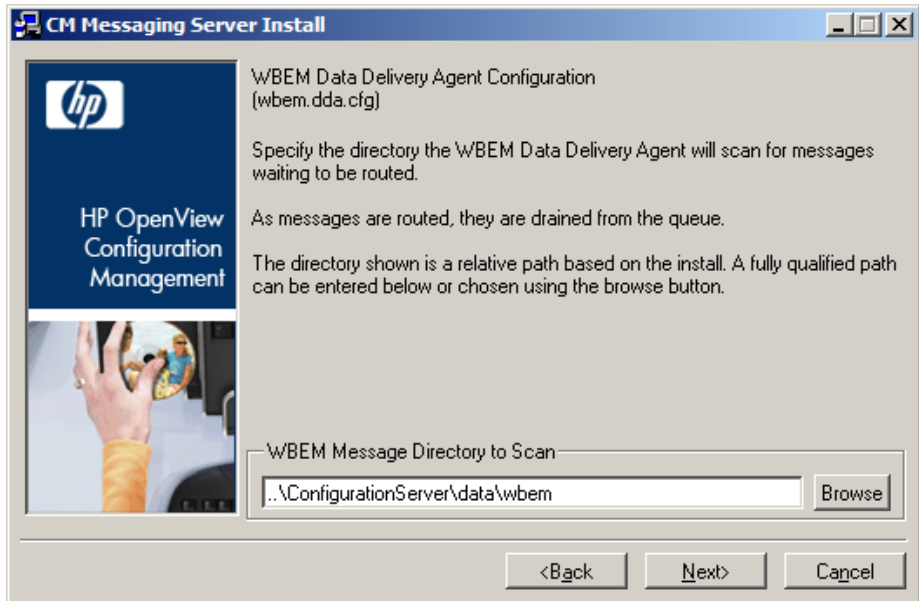
Normally, this is the `\data\inventory` folder located where the CM Configuration Server is installed. This directory will be created if it doesn't already exist when the CM Messaging Server starts. Changes to this directory name must be coordinated with changes to ZTASKEND REXX for a CM Messaging Server co-resident with the CM Configuration Server.

> If necessary, adjust the directory path to your CM 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.

17 Accept the default location, or click **Browse** to select the directory where the Patch Data Delivery Agent should scan for any Patch messages.
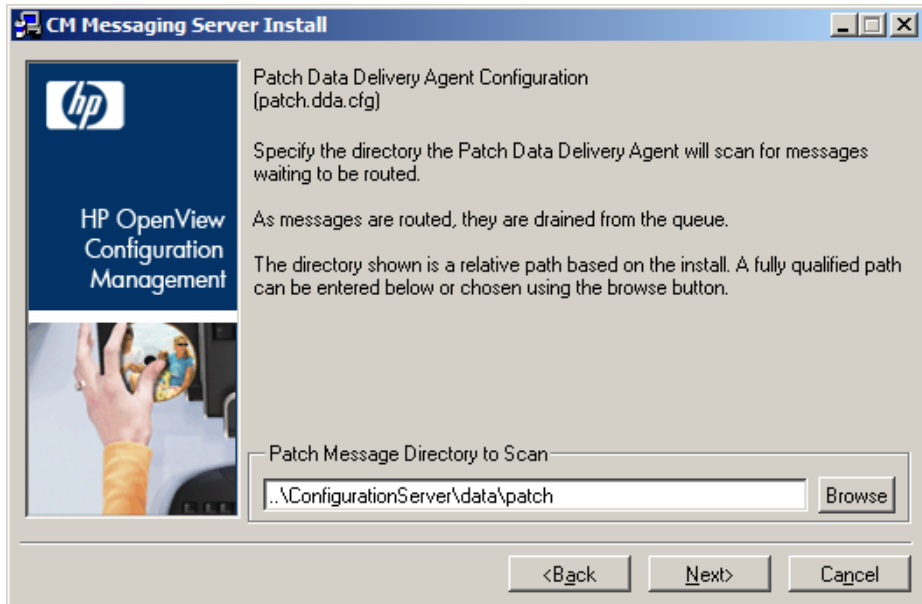
Normally, this is the \data\patch folder located where the CM Configuration Server is installed. This directory will be created if it doesn't already exist when the CM Messaging Server starts up.

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

> For a CM Messaging Server co-resident with the CM 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 60.

The first RMS Configuration window opens.

Task 4   Configure the CM Messaging Server Store and Forward Port

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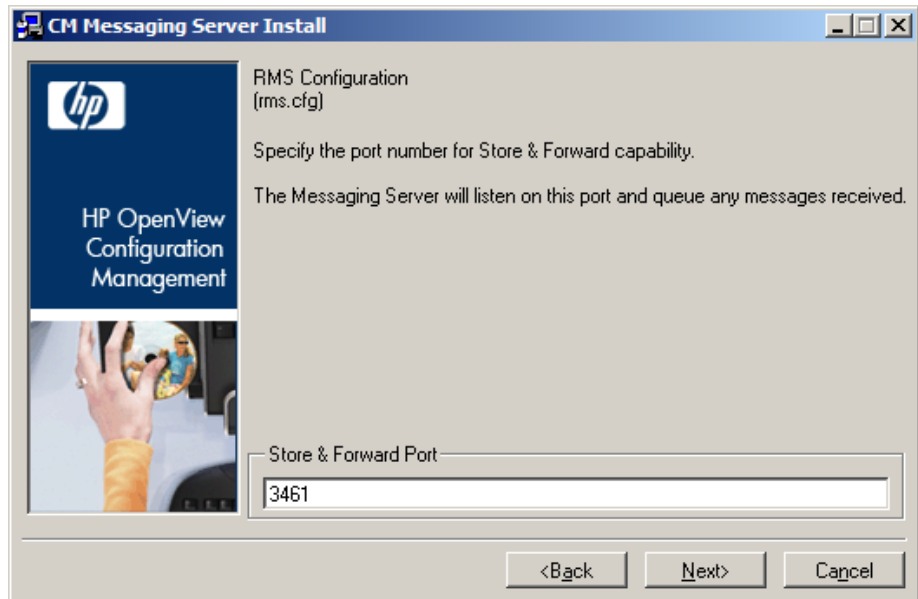
The CM Messaging Server includes the ability to receive and process messages that have been forwarded from other CM Messaging Servers in your enterprise. The port used to receive these messages is called the Store & Forward port.

For more information on using store & forward, see Example 1: Configuring the CM Messaging Server for Store and Forward on page 114.

18 Accept the default store & forward port, 3461, or type another port number to use to receive any messages from other CM 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 a CM
>   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.
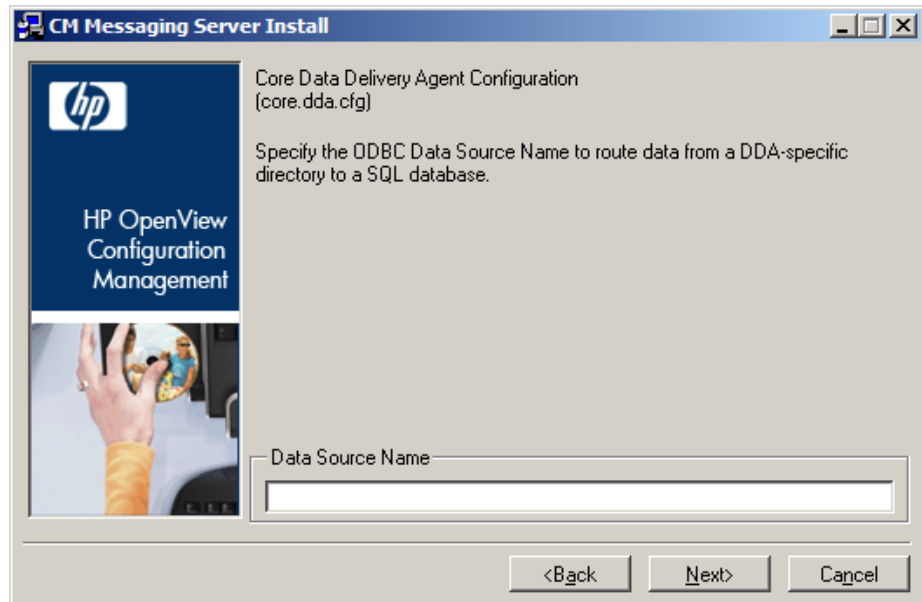
20 Type the Data Source Name of the ODBC SQL database for routing Core
message data.

21 Click **Next**.

The Core Data Delivery Agent Configuration (`core.dda.cfg`) window
opens for the DSN User Name for routing Core data.
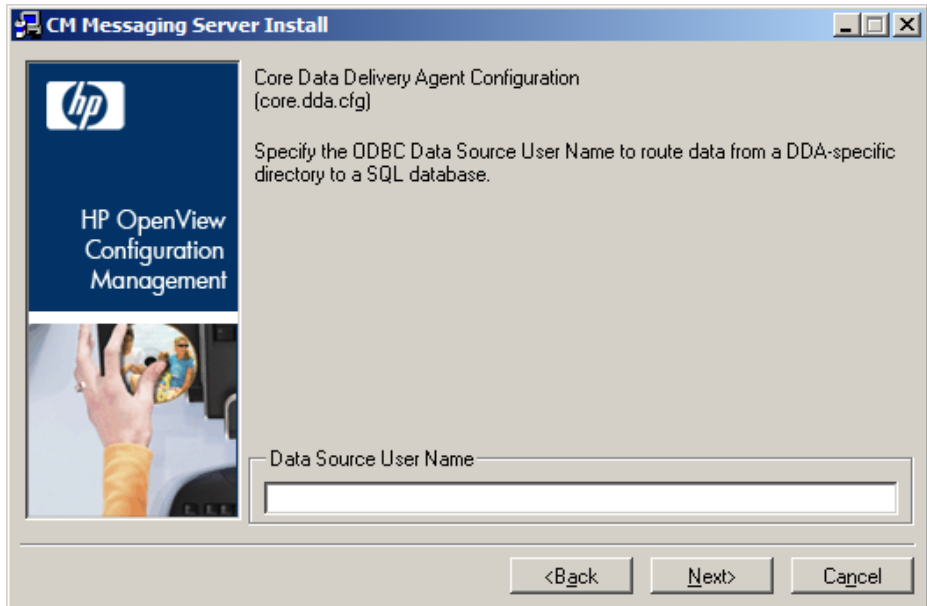
22 Type the DSN User Name to use to connect to an ODBC SQL database for Core data.

23 Click **Next**.

The Core Data Delivery Agent Configuration (core.dda.cfg) window opens for the Data Source Password for routing Core data.
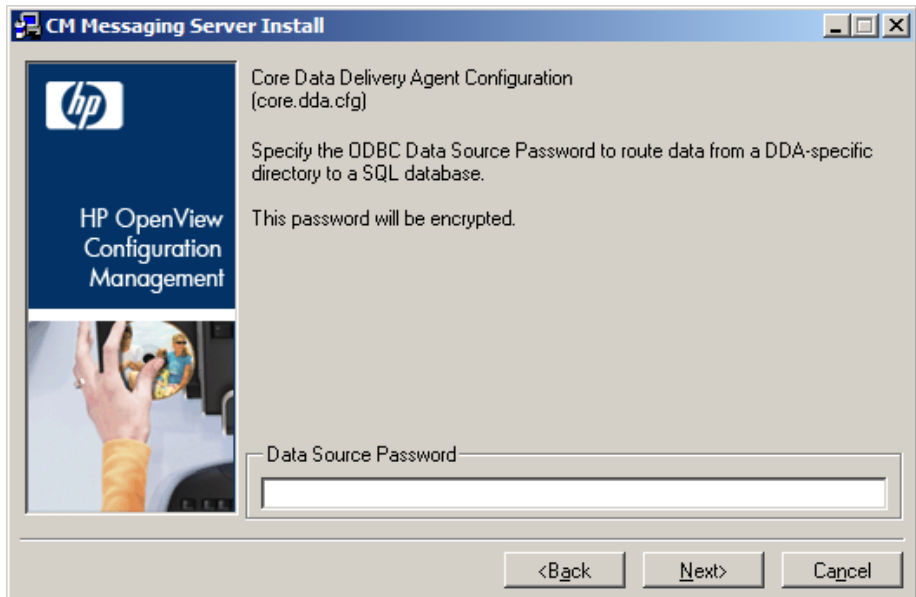
24 Type the password required for the DSN user entered on the previous
    window.

    ▶ The password will be encrypted.

25 Click **Next**.

    The Core Data Delivery Agent Configuration Settings for the CM Portal
    window opens.
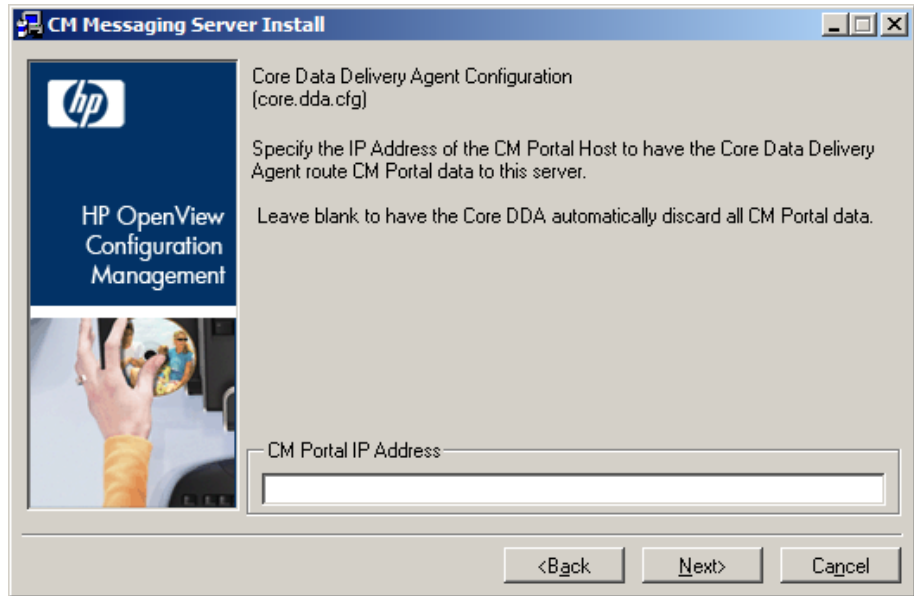
26 Type the IP address or DNS host name of the CM Portal server to route
    all CM Portal data found in the Core DDA scan directory location to that
    server.

    or

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

    ▶ The CM 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 CM Portal.

"Core Data Delivery Agent Configuration (core.dda.cfg)

Specify the port number of the CM Portal Server.

Leave blank if the CM Portal Server IP address was blank."

Below is a field labeled "CM Portal Port" with an empty text box, and buttons at the bottom: <Back, Next>, and Cancel.
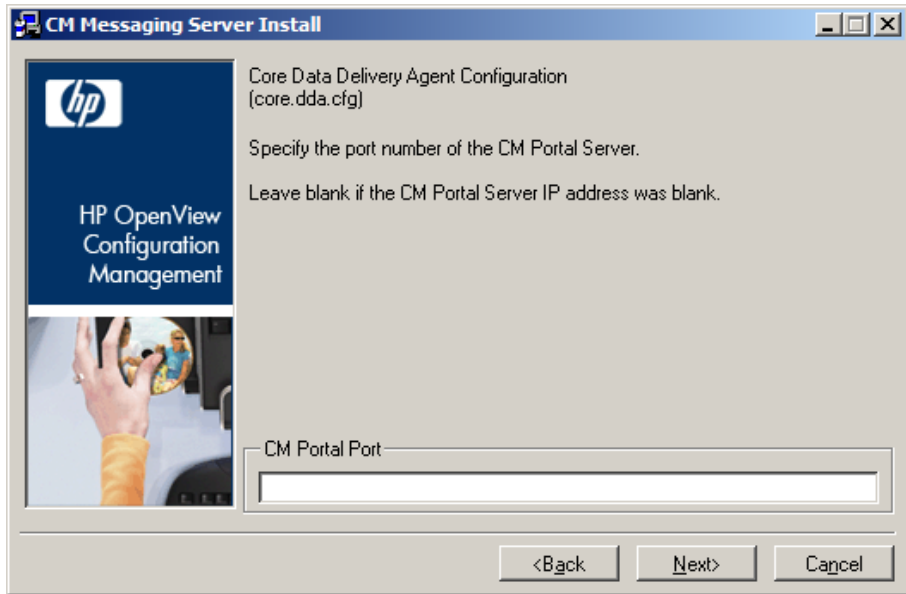
28 If you entered a CM Portal IP Address on the previous window, type the port number of the CM Portal. Normally, this is 3471.

or

Leave the CM Portal Port field blank if you also left the CM 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.

CM Messaging Server Install

HP OpenView
Configuration
Management

Inventory Data Delivery Agent Configuration
(inventory.dda.cfg)

Specify the ODBC Data Source Name to route Filepost Object data from a
DDA-specific directory to a SQL database.

Data Source Name
[   ]

[<Back] [Next>] [Cancel]

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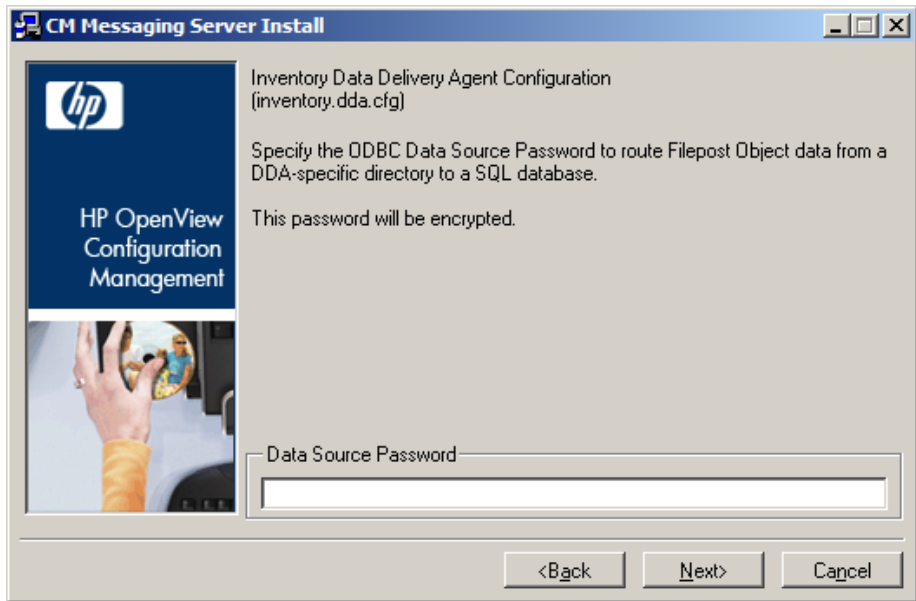
34 Type the password required for the DSN user entered on the previous window.

> The password will be encrypted.

35 Click **Next**.

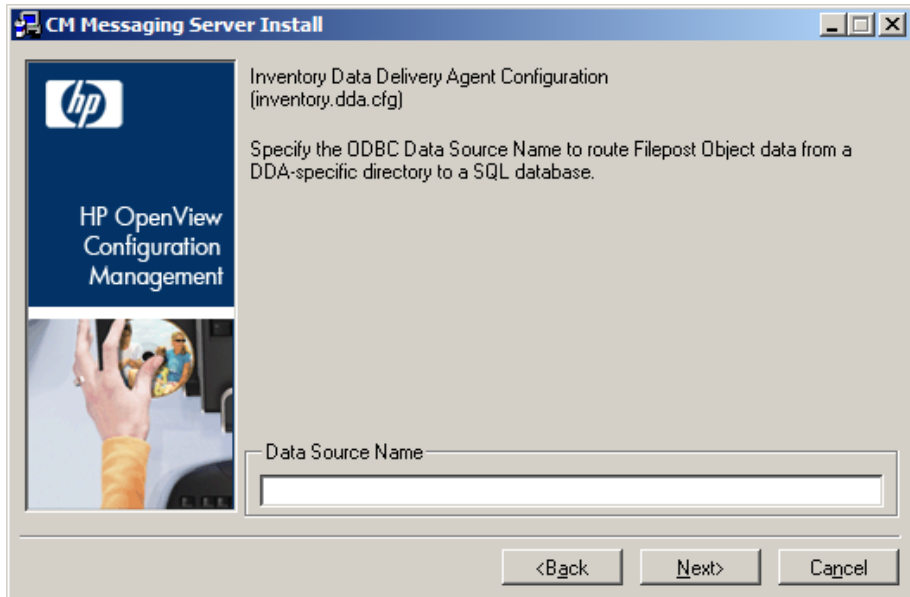**Task 7**   Configure the Wbem Data Delivery Agent (wbem.dda.cfg)

> 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.

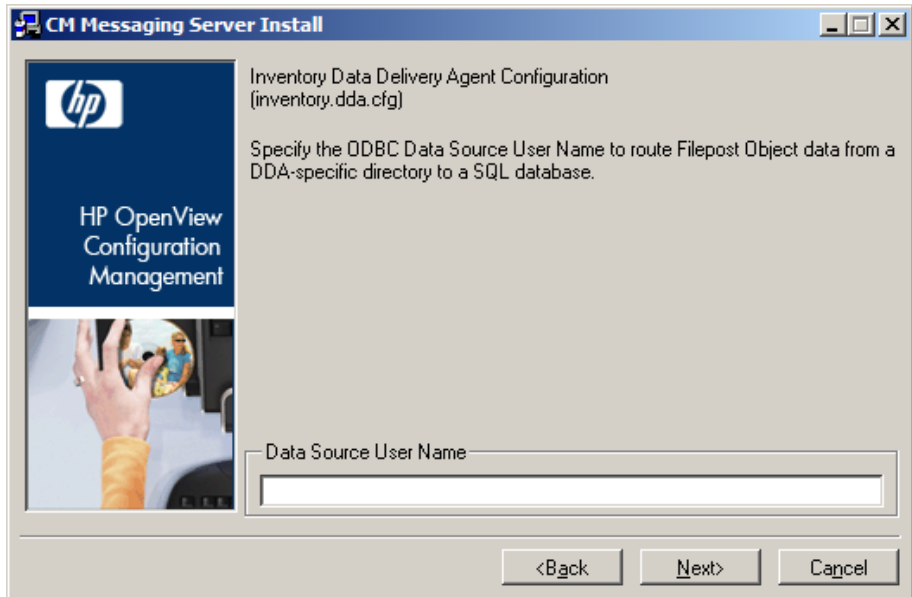Specify the ODBC settings to connect to the SQL database for posting Wbem data in the next three windows.

The Wbem Data Delivery Agent Configuration (wbem.dda.cfg) window opens for the Data Source Name for routing Wbem data.
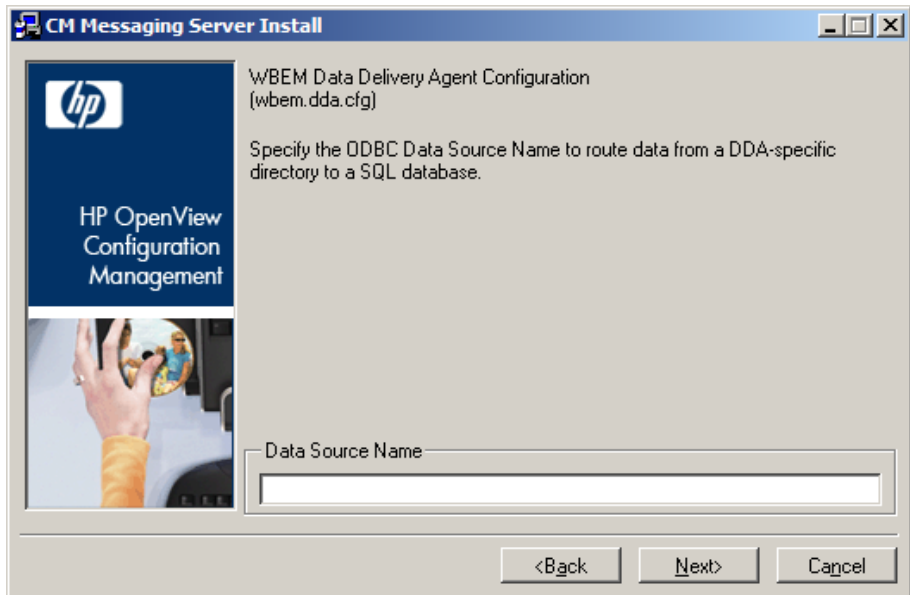
36   Type the Data Source Name of the ODBC SQL database for Wbem data.

37   Click **Next**.

The Wbem Data Delivery Agent Configuration (wbem.dda.cfg) window opens for the DSN User Name for routing Wbem data.
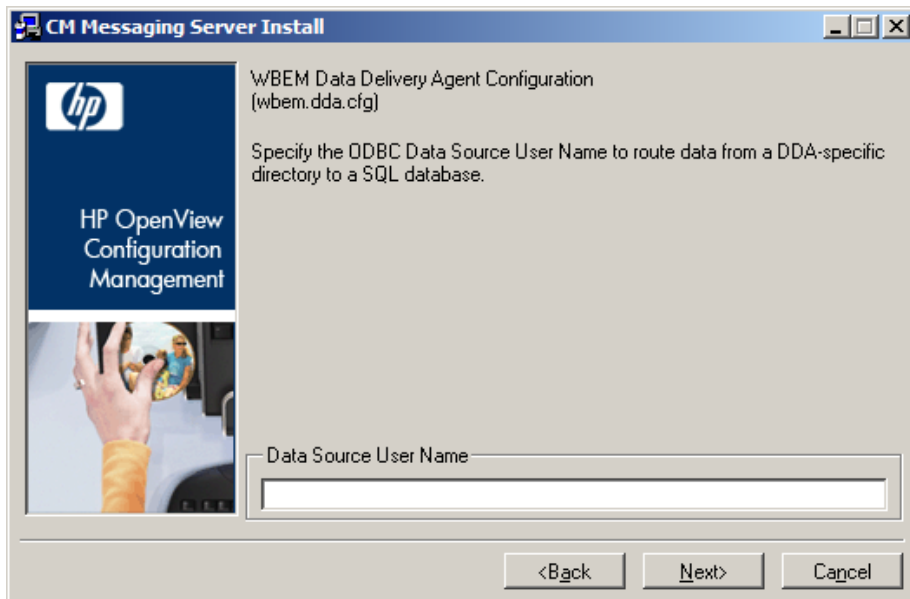
38 Type the DSN User Name to use to connect to an ODBC SQL database for Wbem data.

39 Click **Next**.

The Wbem Data Delivery Agent Configuration (`wbem.dda.cfg`) window opens for the Data Source Password for routing Wbem data.
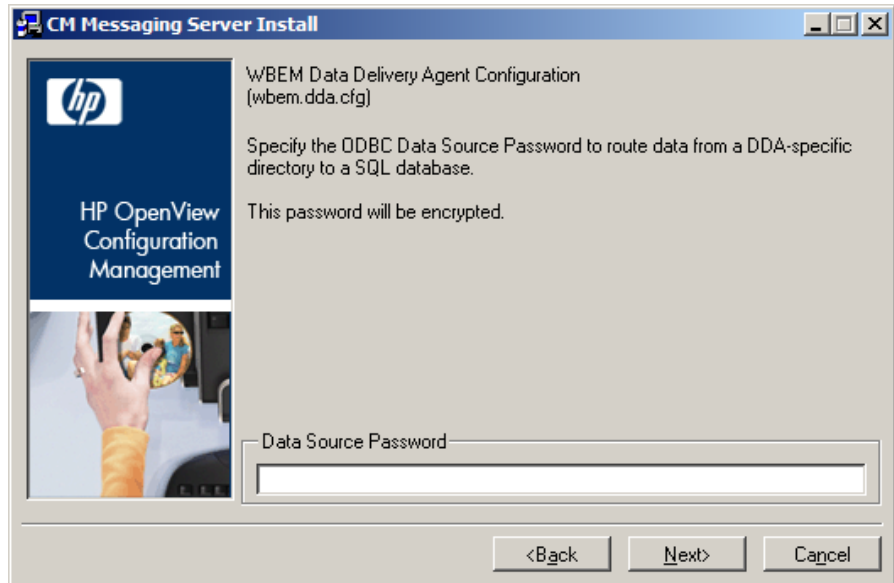
40 Type the password required for the DSN user entered on the previous window.

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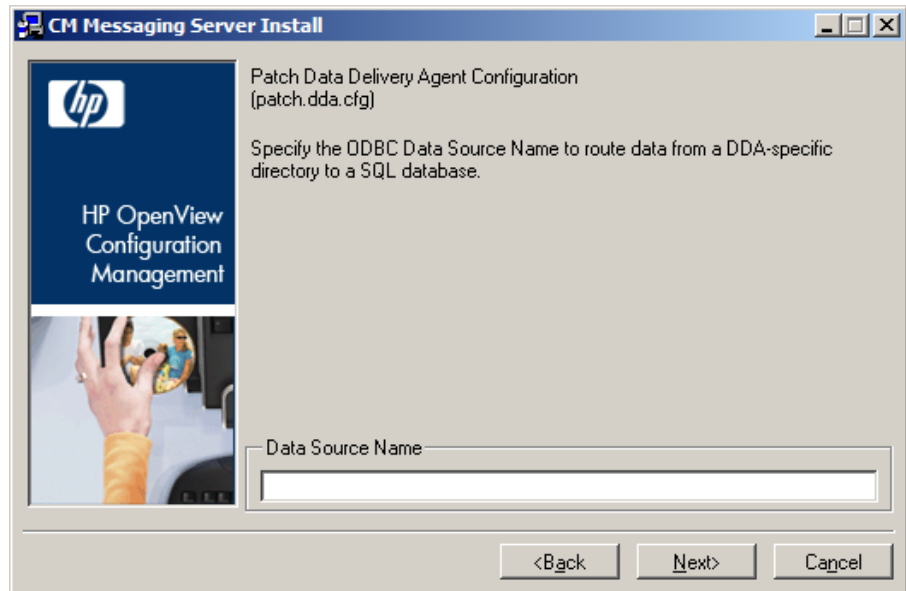
42 Type the Data Source Name of the ODBC SQL database for Patch Manager.

43 Click **Next**.

The Patch Data Delivery Agent Configuration (patch.dda.cfg) window opens for the Data Source User Name for routing Patch data.
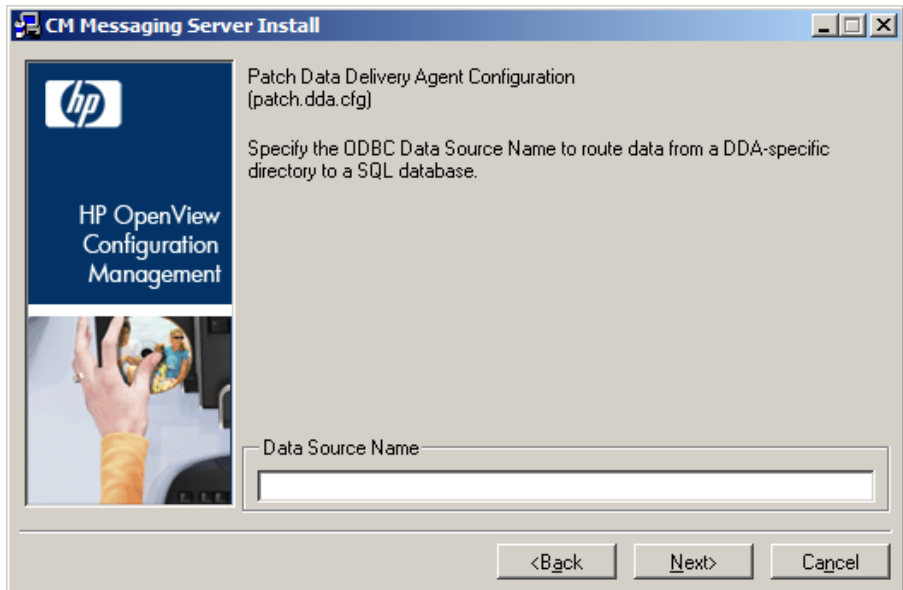
44 Type the Data Source User Name to use to connect to an ODBC SQL database for Patch data.

45 Click **Next**.

The Patch Data Delivery Agent Configuration (patch.dda.cfg) window opens for the Data Source Password for routing Patch data.
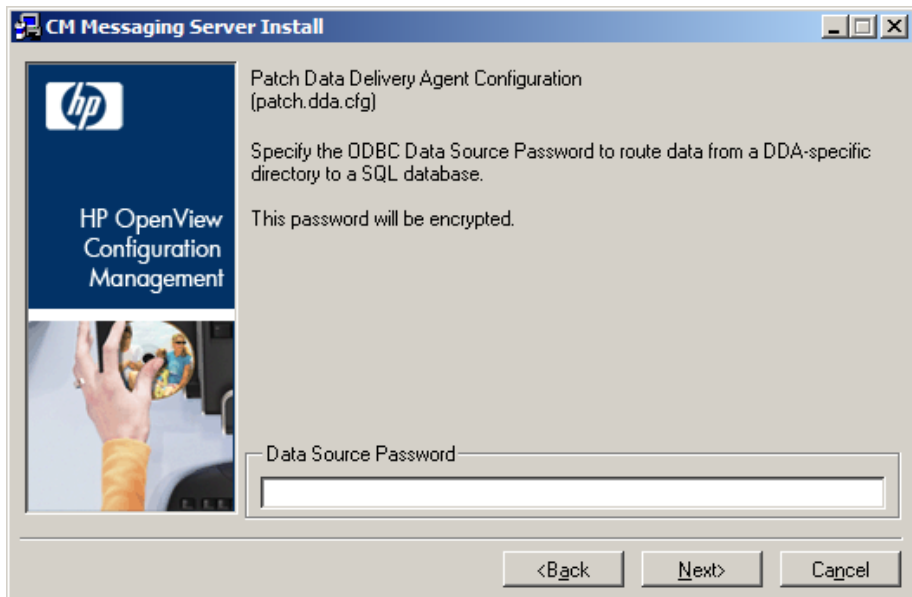
46 Type the password required for the DSN user entered on the previous window.

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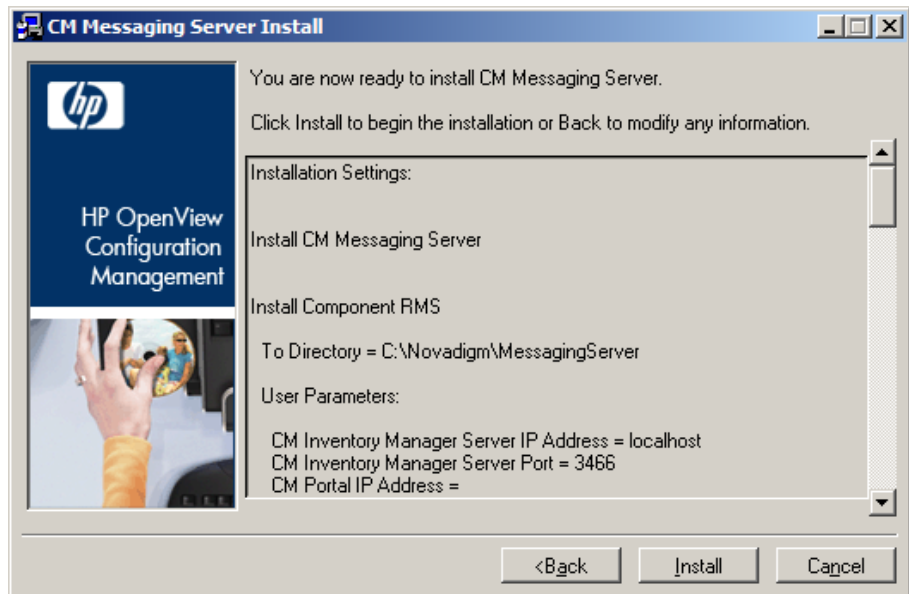
48 Click **Install** to begin the installation.

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

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

The log files for the CM 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 100. 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 CM 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 CM 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:

- `tclodbc` extension needed by the CM 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/nvdmodbc5.20hpux7.tar`

  Target location on Linux:
  `/MessagingServer/odbc/nvdmodbc5.20linux7.tar`

  Target location on Solaris:
  `/MessagingServer/odbc/nvdmodbc5.20solaris7.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 `nvdodbc5.20<platform>7.tar` using chmod command.

   c   Extract the contents of the `.tar` file using the command > `tar -xvf nvdodbc5.20<platform>7.tar`

      Extracting the tar creates the `nvdmodbc` 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 `nvdodbc` directory.

   a   Locate the shell script `nvdmodbc/odbc.sh`

   b   Edit the shell script to point to the lib directory where the ODBC libraries are located: `../nvdodbc/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 [ "$SHLIB_PATH" = "" ]; then
      SHLIB_PATH=/opt/odbc/nvdmodbc/lib
else
      SHLIB_PATH=/opt/odbc/nvdmodbc/lib:$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 nvdodbc/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 `/nvdmodbc` directory located in your Radia Integration Sever 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/nvdmodbc/lib/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=HPCM

Password=HPCM

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:

```
chatr +s enable nvdkit
```

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

```
chatr nvdkit
```

The result of running `chatr 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          "CMPATCH"

    PASS          "<encrypted password>{AES256}"

    DBTYPE        "ORACLE"
```

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

Verify the Patch Method Connections and Queue Name

- CM Patch Manager requires five method connections in the CM Configuration Server Database. For details, refer to the *CM Patch Manager Guide*.

- If you installed the `patch.dda` and changed the name of the Patch Message Directory to Scan value during the CM 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 CM Admin CSDB Editor to edit the ZMTHPRMS attribute of the PRIMARY.SYSTEM. ZMETHOD.PATCH_DEERROR instance, as shown in Figure 3 on page 61.

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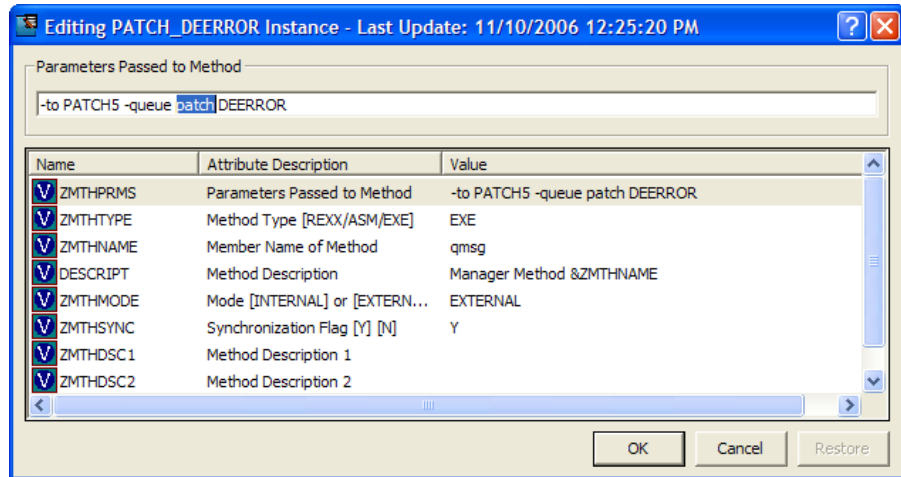
For example: if you entered
`"..\ConfigurationServer\data\mypatch"` as the Patch Directory to
Scan for the `patch.dda`, change the value of ZMTHPRMS in the
PATCH_DEERROR instance from:

```
-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 CM Messaging Server Configuration Parameters are required for
SSL support in a CM Messaging Server RMS.CFG file. Once these

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

1 First update your CM Messaging Server to version 5.x.

2 Refer to the CM Messaging Server topic in the *CM 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 CM 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 CM 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 CM Portal objects to a CM Portal server 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 CM Messaging Server service.

Revert to a CM Messaging Server Configuration for Single-queue Processing

By default, this version of the CM 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 where messages are placed in a single data queue named `default`, and sent to the CM Inventory Manager Server using HTTP (which then posts them to the SQL database), two post-installation changes are required:

- The CM 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 CM Configuration Server `\data\` directory.

For more information, see Processing under Earlier Versions of Messaging Server: 2.x and 3.x on page 76, and Example 2: Configuring the CM Messaging Server to Route Objects from a Single \Data\Default Queue on page 124.

Configure the USAGE.DDA to Aggregate and Forward Usage Data

Following an installation of the CM 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 CM 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 CM Messaging Server is automatically installed as a Windows service. The service name is HP OVCM Messaging Server.

- Use the Services window of your operating system to start or stop the HP OVCM 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 CM 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 CM Configuration Server machine:

  ```
  nvdkit rms.tkd
  ```

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

UNIX Procedures

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

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

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

- To stop the CM Messaging Service, go to the `/MessagingServer` directory on your CM 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.

- To obtain the PID for the CM Messaging Service, check the `rms.log` in the `/logs` directory. The PID is listed with the entry:

  ```
  Messaging Service – main worker started (PID: XXXX).
  ```

  Alternatively, type the following command to list all the UNIX processes for nvdkit:

  ```
  ps -f | grep nvdkit
  ```

  Run the following command to kill the PID listed for the CM Messaging Server – main worker.

  ```
  kill –9 <PID>
  ```

Verify Installation

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

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

  a Look for the entry:

  ```
  Messaging Service started (PID: XXXX)
  ```

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

  ```
  Info: ---------------------------------------------------------------
  Info: Radia Messaging Service (Version 5.00 - Build 105
  Info: Radia Messaging Service - main worker -  started (PID: 1180)
  Info: Platform: Windows_NT
  Info: ------------------------------------------------------------
  ```

  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 5.00 - Build 38.
  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: Radia Messaging Service - coreq worker - started (PID: 2528)
  Info: Platform: Windows_NT
  Info: ---------------------------------------------------------------
  ```

- If the CM Messaging Server has been started as a Windows service, check that the service has been started in the Services Administrative task section of the Control Panel.

- If the CM 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 CM 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 CM Messaging Server base module also has its own worker process. Therefore, if you have installed all possible DDA modules (core, wbem, inventory and patch), there should be six `nvdkit` processes started for the CM Messaging Server.

Summary

- Understand your network topology and have the targets for the CM Messaging Server routes laid out before installing the CM Messaging Server.

- To create a CM Messaging Server environment on your CM Configuration Server, the CM Configuration Server must include a version of ZTASKEND REXX method that calls the QMSG executable for CM Inventory Manager and CM Portal data objects. To route CM Patch Manager objects, the CM Configuration Server must have a method connection to five PATCH_* instances in the PRIMARY.SYSTEM.ZMETHOD class.

- The CM 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 CM Messaging Service is running, it is loading the selected Data Delivery Agent modules, and there is a worker process started for both the CM Messaging Server and each DDA that was loaded.

3 Configuring and Tuning the CM Messaging Server

At the end of this chapter, you will:

- Be able to understand the CM Configuration Server methods that support the CM 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 CM Messaging Server for failover.

> - The first part of this chapter discusses the CM Configuration Server modules that support the CM Messaging Server.
>
> - The topics on configuring and tuning the CM Messaging Server and Data Delivery Agents begin on page 79.
>
> - To configure a CM Messaging Server for Store and Forward, see Example 1: Configuring the CM Messaging Server for Store and Forward on page 114.
>
> - To configure a CM 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 129.
>
> - To configure the Usage Data Delivery Agent, refer to the *CM Application Usage Manager User Guide*.

Understanding the CM Configuration Server Modules that Support the CM Messaging Server

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

Getting Agent information to the CM Messaging Server

Agent objects exchanged with or created on the CM Configuration Server during an agent connect session are formatted into messages for the CM Messaging Server via the CM 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 CM Patch Manager methods. (Refer to Verify the Patch Method Connections and Queue Name on page 60 for details on the CM 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 CM Messaging Server to process the oldest messages first. The CM 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 *HP Configuration Management CM Patch Manager Installation and Configuration Guide (CM 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 CM Inventory Manager, CM Portal, and CM Application Management Profiles, the CM 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 CM 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 QMGS when agent data needs to be collected for another service, such as CM Inventory Manager, CM Application Management Profiles, or the CM Portal. QMSG collects the data and places the messages in specified queues for pickup and processing by the CM 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 CM 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 CM Messaging Server versions 3.x.

Processing Phase-Dependent Objects

Each time a CM agent connects to the CM-CS, 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 CM Inventory Manager (RIM) and CM Portal (RMP). The potential critical objects are:

For RIM: `APPEVENT MSIEVENT SYNOPSIS RNPEVENT`

For RMP: `SYNOPSIS`

In addition to the above critical objects, Table 8 below identifies critical objects collected for each phase.

Table 8   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 9 on page 74 summarizes the Agent Connect phases and the objects collected during the ZTASKEND calls to QMSG, for CORE data going to CM Inventory Manager or the CM Portal.

Table 9   **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
Bootstrap - COP Resolution for Client Operations Profile.	No	Collects these objects for CORE.RIM and CORE.RMP destinations: APPEVENT | MSIEVENT | SYNOPSIS | RNPEVENT SESSION PREFACE ZSTATUS SMINFO
Agent Maintenance Phase	No	Collects these objects for CORE.RIM and CORE.RMP destinations: APPEVENT | MSIEVENT | SYNOPSIS | RNPEVENT SESSION PREFACE ZSTATUS SMINFO

Catalog Resolution: Agent connects to the CM Configuration Server to obtain service resolution list.	**Always.** Collects these objects for CORE.RIM and CORE.RMP destinations:
	  SESSION
	  PREFACE
	  ZCONFIG
	  ZMASTER
	  ZSTATUS
	  SMINFO
		See previous column, but also collects APPEVENT| MSIEVENT | SYNOPSIS | RNPEVENT.

Single Service Resolution: For each service to be resolved, agent makes another connection to the Configuration Server.	No	Collects the following objects for CORE.RIM and CORE.RMP destinations: APPEVENT | MSIEVENT | SYNOPSIS | RNPEVENT SESSION PREFACE ZSTATUS SMINFO

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 CM 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 `CriticalRMPObjects` 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"
CriticalRMPObjects   = "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 CM 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 CM 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 CM 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 CM Configuration Server objects for CM Inventory Manager, CM Portal and CM 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 10 below gives the destination values of QMSG and the configuration file and section used to define its ROUTE.

Table 10   QMSG Destinations and DDA Configuration Locations

QMSG –to destination	Configuration File in *MessagingServer*\etc 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 CM Messaging Server to Route Objects from a Single \Data\Default Queue on page 124.

-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 CM 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 CM Messaging Server processing priority, see the topic How Priority Establishes CM 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 CM 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 CM 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 CM 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 CM 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 CM 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 CM Messaging Server

Use these topics to reconfigure or tune the CM 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 CM Messaging Server and Data Delivery Agents

The CM 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 CM Messaging Server and Data Delivery Agents are found in the\etc directory of where the CM Messaging Server was installed.

All the configuration files for the CM 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 CM Messaging Server and its modules.

A unique identifier follows the msg::register command. Within an instance of the CM Messaging Server, which includes the configuration files for the CM 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 11 on page 81 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 11   Glossary of Section TYPEs and Configurable Parameters

Section TYPE	Configurable Parameters
QUEUE Defines the directory where messages are placed for processing.	**DIR** The directory name of the queue. **USE** Specify the name of the unique identifier that will signify how to dispatch the message. Usually this is a ROUTER type. **POLL** By default, the CM 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. **COUNT** Maximum number of objects to post at a time (during the polling interval defined by POLL). Default is 100 objects. **ATTEMPTS** Maximum number of attempts to retry a failed message delivery before discarding the message. Default is 200 attempts. (By default, the CM Messaging Server and Data Delivery Agents are configured to retry any failed posts every hour, and make up to 200 attempts.) **DELAY** Number of seconds to wait between attempts to retry a failed message delivery. Default is 3600 seconds, or one hour. 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.
ROUTER Defines where the messages are going to sent.	**ROUTE** Delimit each Route by curly braces **TO** This is the address of the message. It is contained in the meta data file of each message when the message is created.

Section TYPE	Configurable Parameters
	USE Specify the unique name of a CM 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).
HTTP This is a way to post the message to another server location using http protocol. The target server can be another CM Messaging Server where the message can be re-queued or it can be another part of the CM infrastructure, such as the CM Portal.	**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. **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. **URL** Specifies the URL to use to send the message.
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.	**ADDRESS** Same as TYPE of HTTP. **PRI** Same as TYPE of HTTP. **URL** Specifies the URL using the https:// convention. Refer to the *HP Configuration Server SSL Implementation Guide* for details.
HTTPD Defines the parameters the CM Messaging Server will use to receive incoming messages.	**PORT** Defines the Port used to "listen" for messages. Only one port specification can be used for a given CM Messaging Server. **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

Section TYPE	Configurable Parameters
	delimit the USE and URL parameters. **USE** Specify the name of the QUEUE type that will receive the incoming messages. **URL** Specify the URL prefix that that will be accepted by the CM 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/**.
ODBC Used to post PATCH messages into a SQL database.	**DSN** Data Source Name **USER** User ID for the DSN **PASS** Password for the DSN
FILTER A means to route PATCH data into multiple SQL tables.	**USE** Specify the name of the unique identifier that will signify how to dispatch the message. **TO** This is the address of the message. It is contained in the meta data file of each message when the message is created. **FILTER** Used for PATCH object processing into multiple tables
COREODBC Used to post CORE messages into a SQL database. Only configurable in `core.dda.cfg` and `inventory.dda.cfg`.	**DSN** Data Source Name **USER** User ID for the DSN **PASS** Password for the DSN **DSN_ATTEMPTS** Number of attempts to connect to the DSN. Default is 1. **DSN_DELAY** Delay in seconds between attempts to connect to the

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

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 CM Messaging Server was installed.

The base CM 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 it 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 86

- About the Sections in the CORE.DDA.CFG File on page 88

- About the Sections in the INVENTORY.DDA.CFG File on page 92

- About the Sections in the WBEM.DDA.CFG File on page 95

- About the Sections in the PATCH.DDA.CFG File on page 97

- Additional Tuning Topics on page 100

> To configure the USAGE.DDA.CFG file, see Configuring Aggregation in the *CM Application Usage Manager User Guide*.

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

1 Stop the CM Messaging Server before editing the `rms.cfg` file. For details, see Starting and Stopping the CM Messaging Server on page 64.

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 CM Messaging Server.

About the Sections in the RMS.CFG File

The CM 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 (CM Inventory Manager and CM Portal objects), wbem data (wbem audit data for CM Inventory Manager) and patch data (for CM Patch Manager)

Optional entries in the `rms.cfg` file can include SSL support, a "`msg:register httpd`" section if this CM Messaging Server is receiving forwarded messages from another CM Messaging Server, and a "`msg:register default`" section if this CM 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 CM 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 *CM SSL Implementation Guide.*

- **log::init**
  Sets the Logging Level for entries written to the log files. The default is 3. Normally, this is not changed. For details on changing the logging level, see Configuring the Log Level, Log Size and Number on page 101. The log files are located in the Logs directory of where the CM Messaging Server was installed.

- **log.configure –stderr 0**
  Required by the CM Messaging Server log. Do not modify.

- **log.configure -lines 50000**
  The default number of lines contained in a log before the log is rolled over.

- **log.configure -limit 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 CM Portal directory. See About the Sections in the CORE.DDA.CFG File on page 88 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 92 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 95 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 97 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 CM 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 CM Messaging Server to Route Objects from a Single \Data\Default Queue on page 124.

  > 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 101 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 CM Messaging Server co-located with a CM 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 CM 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 CM 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 CM Portal can be routed using HTTP to a CM Portal Zone, or discarded using the built-in `/dev/null` location.

- Core object data can be routed using HTTP to an existing CM Inventory Manager Server. This option is no longer supported in CM Messaging Server v5, as the CM 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 CM Messaging Server is receiving messages forwarded from another CM

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 CM Messaging Server handles the messages placed by QMSG in the `/data/core` folders.

  The parameters are summarized below:

  — TYPE of QUEUE defines the CM 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 CM 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 100.

  — 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 CM 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 104.

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

CM Portal directory from the CORE messages being posted to an ODBC database. For more information, see About the CM Portal Data Queue (rmpq) in CORE.DDA.CFG on page 104.

> 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 91.

  > 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 CM 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 100 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 101 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 CM Inventory ODBC database. Enclose the entry in quotes.
USER	Specify the user name for the CM Inventory ODBC database identified in the DSN parameter.
PASS	Specify the password for the user of the CM 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 99.
DSN_ATTEMPTS	Number of attempts to connect to the CM Inventory Manager database DSN. Default is 1.
DSN_DELAY	Delay in seconds between attempts to connect to the CM 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)	In the WBEMODBC section, a switch to enable the creation of a new SQL file and table in the CM Inventory ODBC database when a new object class is received. Default is 0. 0 – Does not create a SQL file or table entry for a new object class. 1 – Creates a new SQL file and table entry for a new object class.
STARTUPLOAD	No longer supported; any coded value is ignored. ▶ As of Version 5.0, the SQL tasks to create the tables and load the scripts for the CM Inventory Database are always performed upon CM Messaging Server and Data Delivery Agent startup.

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 CM 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 CM Messaging Server co-located with a CM 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 CM Messaging Server receiving messages forwarded from another CM 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 CM 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 CM 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 CM Inventory Server can post the messages to the back-end CM 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 CM Messaging Server is receiving messages forwarded from another CM 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 CM Messaging Server handles the messages placed by QMSG in the `/data/inventory` folders.

The parameters are summarized below:

— TYPE of QUEUE defines The CM 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 CM 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 100.

— 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 CM 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 91.

  > 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 CM 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 100 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 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 CM Messaging Server co-located with a CM 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 CM Messaging Server receiving messages forwarded from another CM 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 CM 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 CM
  Messaging Server is receiving messages forwarded from another CM
  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
  CM 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 CM 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 CM 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 100.

  — 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 CM
  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 91.

- **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 CM Messaging Server when using store and forward, or a CM 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 CM 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 100 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 101 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 CM 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 CM

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 CM 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 CM 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 100.

— 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 on page 99.

  > 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 101 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 CM Patch Manager ODBC database. Enclose the entry in quotes.
USER	Specify the user name for the CM 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 CM 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 CM 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 CM 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 CM Messaging Server and the Data Delivery Agents are configured to retry any message that fails to post. By default, the CM 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 12 on page 106 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 CM 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 CM 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 CM 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 CM 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. 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 CM 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 CM 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 CM Messaging Server `\logs` directory.

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

```
Log.configure -stderr 0
```

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:

```
log.configure -lines maximum_lines
log.configure -size 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 a `core.dda.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
}

log.configure -lines 1000
log.configure -limit 10

# ATTEMPTS * DELAY = Maximum time in seconds an item will remain
in the queue
# 200 * 3600 = ~8 days
```

Configuring the CM Messaging Server to Discard or Drain Messages

The location of /dev/null is built into the CM Messaging Server for discarding messages. When the USE parameter is set to /dev/null in any of the ROUTER type sections of the CM Messaging Server or Data Delivery Agent configuration files, the messages being processed will be successfully discarded without an error.

Example: Discarding messages for the CM 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 12 on page 106 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 CM Messaging Server to Route CM Portal Messages

About the CM 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 CM Portal directory into its own queue, named rmpq. This allows for separate throttling of the CORE messages being posted via HTTP to the CM 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 CM Portal Messages

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

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

1   Use a text editor to edit the `core.dda.cfg file`, located in the `etc` folder of the CM 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
```

4   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
```

5   Edit the URL value of `localhost:3466` to indicate the host and port of your CM Portal server. For example, a CM 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.

6   Save your changes and restart the CM Messaging Server. For details, see Starting and Stopping the CM Messaging Server on page 64.

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 CM 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 12   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
\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 CM Messaging Server or Data Delivery Agent processing the queue to be disabled. Table 12 above identifies the configuration file to use for each queue type.

2   Locate the 'msg::register' section named in Table 12; 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 CM 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 CM Messaging Server Processing Order on page on
page 78.

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 CM Messaging Server

The CM 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 CM 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 CM 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 76.

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 CM 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 CM 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 CM 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 CM 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 CM Messaging Server Configurations

The CM 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 CM Messaging Server in alternative configurations. These configurations include:

- Example 1: Configuring the CM Messaging Server for Store and Forward on page 114.

- Example 2: Configuring the CM Messaging Server to Route Objects from a Single `\data\default` Queue.

- Example 3: Configuring CM Messaging Server to Route to Multiple Queues on page 127.

> 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 CM 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 CM Messaging Server for Store and Forward

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

Figure 4   Store and Forward "Hop"
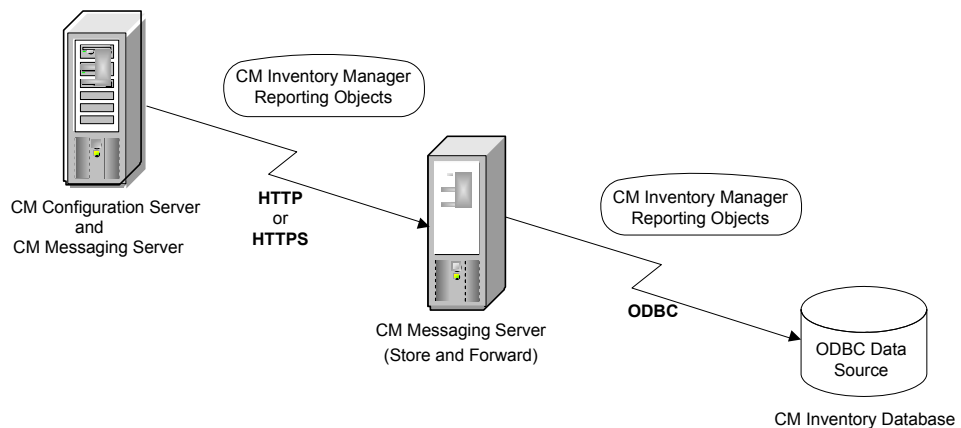
The concept of Store and Forward is moving the messages through the network to a location closer to where the work will be done on them. The messages in essence "hop" from CM Messaging Server to CM Messaging Server. The forwarding CM Messaging Servers route data using HTTP or HTTPS.

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 CM Messaging Server on the CM Configuration Server to forward all inventory messages to another, remote, CM Messaging Server. This configuration drains the inventory messages from the CM Configuration Server location quickly, freeing up CM 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 115.

- Configuring a CM Messaging Server to Forward Messages begins on page 118.

Installing and Configuring a "Receiving" CM 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 CM Messaging Server to CM Messaging Server. The downstream CM 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" CM Messaging Server from a "Sending" CM 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 CM-CS is hosted), of any intermediate CM Messaging servers, and of the final destination CM 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 CM Portal.

About the CM Messaging Server Receiver

The CM 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 CM 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 CM 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 CM 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 CM 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 CM Messaging Server queues messages sent to it in the newly created "Message Directory to Scan" location. The CM 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 59 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 CM Messaging Server on page 69.

Configuring a CM Messaging Server to Forward Messages

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

Let us assume the CM Messaging Server was installed with DDAs for each of the CM Inventory Manager objects (CORE, INVENTORY, and WBEM DDAs), and we want to forward those messages to another CM Messaging Server installed with those DDAs listening downstream. We want the DDAs

on the downstream CM 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 CM 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 CM Messaging Server to forward CORE, INVENTORY and WBEM messages

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

The procedures to forward PATCH messages are similar. They begin on page 122.

- 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 CM 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 79.

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 13 on page 121 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 CM Messaging Server and Port named in the URL. The next step is to replace the default values in the URL to specify the downstream CM 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 CM Messaging Server and port number. The default Store and Forward *port* number for the CM Messaging Server is 3461.

Table 13 on page 121 shows the changes made to the coreforward and corerouter sections to forword 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 CM 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 13 on page 121 shows the `msg:register coreodbc` section commented out.

Table 13   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 CM 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 CM Messaging Server,
    normally 3461.

7   Save the changes to the configuration files that were edited. On the
    downstream server hosting the CM 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 CM Messaging
    Server Receiver on page 115.

8   Restart the CM Messaging Server (rms) Service. For details, see Starting
    and Stopping the CM Messaging Server on page 64.

Forwarding PATCH Messages

To configure a CM Messaging Server to forward PATCH messages to another CM 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 115. 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 CM 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 CM Messaging Server to receive the patch massages. 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 CM Messaging Server, normally 3461.

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

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

Example 2: Configuring the CM 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 CM 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 CM 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 CM 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 100.

  — 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 100.

  — Configure the section to Discard Messages. See Configuring the CM Messaging Server to Discard or Drain Messages on page 103.

- (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 101 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 14   RMS.CFG Parameters used to Define the /Data/Default Queue

Parameter	Default	Definition
TYPE	QUEUE	Registration type. *Do not change this value.*
DIR	../ConfigurationSer ver/data/default	Directory where your CM Configuration Server (through QMSG.exe) will queue XML objects to post. Edit the DIR value to reflect the full path of your data/default directory *using forward slashes* for both Windows and UNIX platforms.
USE	router	Internal setting telling the program what process to use. *Do not change this setting.*
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 CM Messaging Server will try to post a message before discarding it. Note: DELAY * ATTEMPTS 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 CM Messaging Server configuration, with each worker performing a separate POLL and

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

Example 3: Configuring CM Messaging Server to Route to Multiple Queues

This example configures the CM 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 128, 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 CM 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 DNSs 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 92.

3 Processing of the second queue basically duplicates the first:

   a A QUEUE for `coreodbq2` 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.

B Product Name Changes

If you have used Radia in the past, and are not yet familiar with the newly rebranded HP terms and product names, Table 15 below will help you identify naming changes that have been applied to the Radia brand.

Table 15   Product Name and Term Changes

New Name/Term	Old Name/Term
CM agents	Radia clients
HP Configuration Management	Radia
HP Configuration Management Admin CSDB Editor	Radia System Explorer
HP Configuration Management Application Management Profiles	Radia Application Management Profiles
HP Configuration Management Configuration Server	Radia Configuration Server
HP Configuration Management Configuration Server Database	Configuration Server Database, Radia Database
HP Configuration Management Inventory Manager	Radia Inventory Manager
HP Configuration Management Messaging Server	Radia Messaging Server
HP Configuration Management Portal	Radia Management Portal
HP Configuration Management Patch Manager	Radia Patch Manager
HP Configuration Management Solutions for Servers	Server Management

Index