

HP OpenView Messaging Server Using Radia

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

Software Version: 3.2

Installation and Configuration Guide

*Information in this guide can be used for all supported platforms except where indicated for a specific platform only.

Document Release Date: February 2006



Legal Notices

Warranty

Hewlett-Packard makes no warranty of any kind with regard to this document, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Hewlett-Packard shall not be held liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.

A copy of the specific warranty terms applicable to your Hewlett-Packard product can be obtained from your local Sales and Service Office.

Restricted Rights Legend

Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause in DFARS 252.227-7013.

Hewlett-Packard Company
United States of America

Rights for non-DOD U.S. Government Departments and Agencies are as set forth in FAR 52.227-19(c)(1,2).

Copyright Notices

© Copyright 1998-2006 Hewlett-Packard Development Company, L.P.

No part of this document may be copied, reproduced, or translated into another language without the prior written consent of Hewlett-Packard Company. The information contained in this material is subject to change without notice.

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

This manual's title page contains the following identifying information:

- Version number, which indicates the software version
- 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://ovweb.external.hp.com/lpe/doc_serv/

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 since the last released edition.

Table 1 Document Changes

Chapter	Version	Changes
Chapter 1	3.0	Page 16, Features and Capabilities added the ability to "Maintain multiple data queues on Store and Forward Messaging Servers."
Chapter 1	3.1	Page 16, Benefits over Previous Implementations: Version 3.1 adds new benefits, including: <ul style="list-style-type: none">• Configurable control over when SQL tables are created and commands are generated—either upon <i>first message delivery</i> (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 16, 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	3.0	Page 19, 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 20, 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 21, 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 22, 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 27, Tips: new topic.
Chapter 2	3.0	Page 28, 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 28 to guide you through the install.
Chapter 2	3.1	Page 32, Table 4, new table summarizes which DDAs to install by product.
Chapter 2	3.0	Page 61, 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 64, Reconfiguring the Messaging Server for RMS 2.x 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	Version	Changes
Chapter 2	3.1	Page 67, Verify Installation: expanded topic discusses how to use the <code>rms.log</code> to verify the Messaging Server and Data Delivery Agents are running, and the <code>nvdkit</code> processes are available for each queue worker.
Chapter 3	3.0	Page 75, About the Patch Method for Collection75, new topic to explain the method used to call QMSG for Patch data.
Chapter 3	3.0	Page 75, 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 84, Configuring the Messaging Server: contents are changed substantially due to the loading of data delivery agent modules. Very few items are configured directly in the <code>rms.cfg</code> file as of version 3.0.
Chapter 3	3.0	Page 85, Table 8 gives a glossary of all configurable section TYPES and their parameters.
Chapter 3	3.1	Page 88, Table 8, 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 88, Table 8, a new AUTOCREATE configurable parameter was added for the WBEMODBC section type.
Chapter 3	3.0	Page 93, 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 91, Additional Sections in the RMS.CFG File, entries for log.configure -limit and log.configure -lines were added to this topic.

Chapter	Version	Changes
Chapter 3	3.1	Page 94, 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 96, ODBC Settings for CORE, INVENTORY and WBEM Objects, new STARTUPLOAD and AUTOCREATE configuration parameters were added to this topic.
Chapter 3	3.0	Page 98, 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 100, 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 103, 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 106, 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 110, About the Management 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.
Appendix A	3.0	Page on page 120, Example 1: Configuring the 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.

Chapter	Version	Changes
Appendix A	3.0	<p>Page 120, Example 2: Configuring the Messaging Server to Route Objects from a Single \Data\Default Queue, new topic explains how to use sections in the <code>rms.cfg</code> file to route data objects placed in a single <code>data\default</code> queue, when Data Directory Agents are <i>not</i> used.</p> <p>Note: This topic was previously discussed in Configuring the Messaging Server. It was relocated due to the use of multiple data queues as of Messaging Server v 3.0.</p>
Appendix A	3.0	<p>The earlier configuration example: <i>Configuring the Messaging Server for Custom Named Queues</i> has been deleted. The adoption of multiple Data Delivery Agents with individual queue names has eliminated the need for this customization.</p>
Appendix A	3.0	<p>Page 136, Example 4: Configuring Data Delivery Agents to Route to Multiple DSNs using ODBC, new example illustrates how to customize a <code>core.dda.cfg</code> file to create two queues to route messages to two separate DSNs. This example duplicates message delivery of <code>CORE.ODBC</code> messages to two different target databases.</p>



The chapter *Migrating Inventory Processing to Use QMSG and the Messaging Server* has been deleted from this guide. For migration information, refer to the *Upgrade Procedures Guide for the HP OpenView Messaging Server 3.0 using Radia*. This guide is located in the `migrate` folder of where the Messaging Server is located on the Radia Infrastructure CD.

Support

Please visit the HP OpenView support web site at:

<http://www.hp.com/managementsoftware/support>

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

HP OpenView 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 valuable support customer, you can benefit by using the support site to:

- Search for knowledge documents of interest
- Submit enhancement requests online
- Download software patches
- Submit and track progress on support cases
- Manage a support contract
- Look up HP support contacts
- Review information about available services
- Enter 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 log in. Many also require a support contract.

To find more information about access levels, go to:

http://www.hp.com/managementsoftware/access_level

To register for an HP Passport ID, go to:

<http://www.managementsoftware.hp.com/passport-registration.html>

Contents

1	Introduction	15
	About the Messaging Server	16
	Features and Capabilities	16
	Benefits over Previous Implementations.....	16
	Messaging Server Processing on the Configuration Server	17
	About the Data Delivery Agents	19
	About Routing Inventory Data	20
	About the SQL Database Tables and Scripts	21
	About Using Store and Forward	22
	Summary	24
2	Installing the Messaging Server	25
	Messaging Server Installation	26
	Platform Coverage	26
	Tips.....	27
	Tips for Installing Data Delivery Agents.....	28
	Installation Procedures for Windows and UNIX	28
	Overview of Installation Tasks.....	28
	Verify the Patch Method Connection and Queue Name.....	61
	Enabling HTTPS Routing using SSL	62
	Reconfiguring the Messaging Server for RMS 2.x Processing	64
	Starting and Stopping the Messaging Server.....	65
	Windows Procedures	65
	UNIX Procedures	65
	Verify Installation	67
	Summary	69
3	Configuring and Tuning the Messaging Server	71
	Understanding the Configuration Server Modules that Support the Messaging Server ..	72

Getting Client information to the Messaging Server (RMS)	72
About the Patch Method for Collection.....	75
About the ZTASKEND REXX method.....	75
ZTASKEND calls to QMSG	76
Processing Phase-Dependent Objects	76
Processing Always Objects.....	79
Processing under RMS Version 2.x and RMS Version 3.x.....	80
QMSG Method Syntax.....	80
How Priority Establishes Messaging Server Processing Order	82
Configuring the Messaging Server	84
Editing the Configuration Files for the Messaging Server and Data Delivery Agents	84
ODBC Settings for CORE, INVENTORY and WBEM Objects	96
ODBC Settings for PATCH Objects	104
Additional Tuning Topics.....	106
Configuring the Poll Interval and Post Quantity	106
Configuring for Retry Attempts	106
Configuring for Failover	106
Configuring the Log Level, Log Size and Number.....	107
Changing the Logging Level.....	108
Changing the Size and Number of Log Files	108
Configuring the Messaging Server to Discard or Drain Messages	109
Configuring the Messaging Server to Route RMP Messages.....	110
About the Management Portal Data Queue (rmpq) in CORE.DDA.CFG.....	110
Restoring Routing for Management Portal Messages.....	111
Disabling Processing of Messages in a Queue	112
Modifying the Priority in which Messages are Processed	113
4 Troubleshooting	115
Troubleshooting the Messaging Server.....	116
Problem: Log indicates no route defined or failed delivery	116
Solution:.....	116
Problem: Error 404 or 500	116
Solution:.....	117
Summary.....	118

A	Optional Messaging Server Configurations.....	119
	Example 1: Configuring the Messaging Server for Store and Forward.....	120
	Installing and Configuring a "Receiving" Messaging Server	121
	About the RMS Receiver.....	122
	Configuring the Receivers for the .dda Modules	122
	Running the Installation for a Receiving Server and DDAs.....	124
	Configuring a Messaging Server to Forward Messages to another Messaging Server.....	125
	About Forwarding Messages to a Receiving Messaging Server or DDA.....	125
	Example 2: Configuring the Messaging Server to Route Objects from a Single	
	\Data\Default Queue.....	129
	Configuring the Register Default Section	131
	Example 3: Configuring Messaging Server to Route to Multiple Queues	133
	Example 4: Configuring Data Delivery Agents to Route to Multiple DSNs using ODBC.....	136
	Index	141

1 Introduction

At the end of this chapter, you will:

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

About the Messaging Server

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

On an HP OpenView Configuration Server using Radia (Configuration Server), the Messaging Server operates hand-in-hand with the executable, QMSG, to handle the transfer of data reported from Radia clients to the appropriate ODBC reporting database or external Radia Server.

Features and Capabilities

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

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

Benefits over Previous Implementations

- Multiple, specialized queues allow separate workers to operate on each queue and allows for parallel processing of object messages.
- Independent data delivery agents allow for modular upgrades.
- Using the Messaging Server to post object data directly to a SQL database via ODBC can eliminate the need for the Inventory Manager Server.

- Eliminates bottlenecks on the Configuration Server caused by the back-up of processing of reporting data on the Inventory Server.
- Reliability of processing Inventory and Patch data is maintained, due to:
 - Built-in retry capability.
 - Ability to reroute messages remaining in a queue to a failover location.
 - Retry, holding, and re-routing features eliminate potential loss of data caused by network failures.
- Improved efficiency and control over when SQL tables are created and commands are loaded. A STARTUPLOAD parameter can have these tasks performed upon first message delivery (HP recommended option and the default) or upon Messaging Server startup.
- Improved queue control and throttling capability for posting Management Portal data.
- New support for customized message routing to multiple DSNs using ODBC.

Messaging Server Processing on the Configuration Server

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

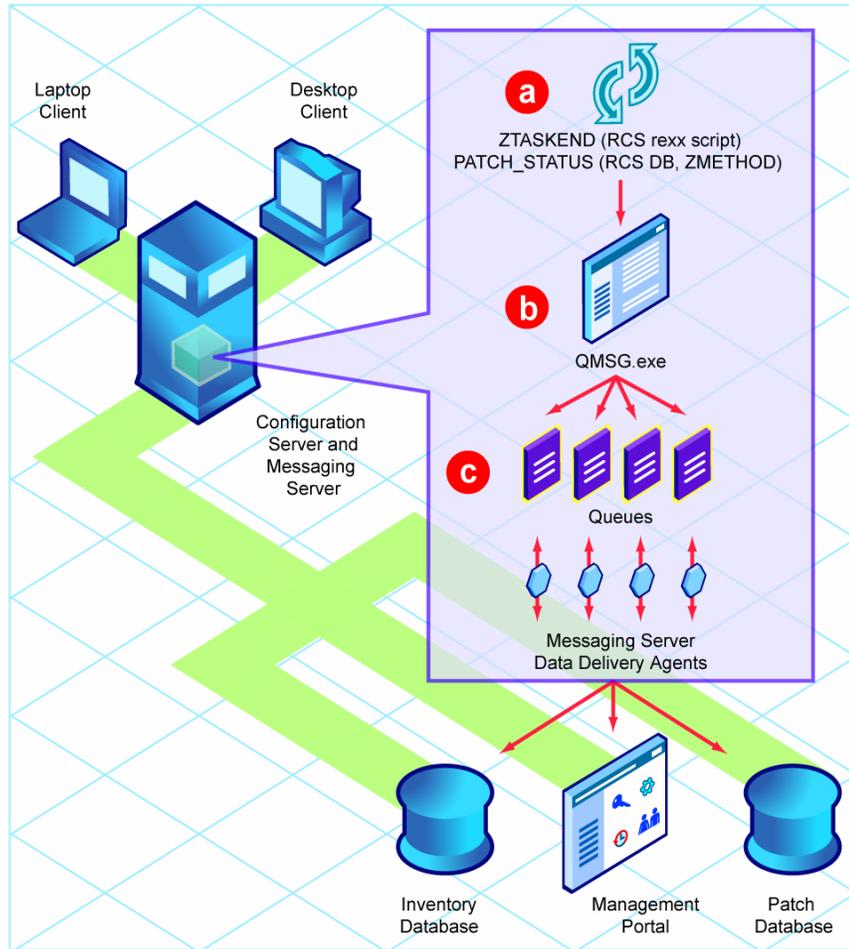
When a customer has multiple Configuration Servers, each one will have a co-located Messaging Server and the ability to route object data to the appropriate target location.

Figure 1 on page 18 provides an overview of Messaging Server Processing.

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

The Messaging Server refers to CORE objects as the standard Radia client objects exchanged. Examples of the CORE objects are ZMASTER, PREFACE and SESSION. Other types of objects that can be exchanged are multi-heap WBEM reporting objects, client FILEPOST objects created by a client inventory audit process (called INVENTORY objects) and the ZOBJSTAT and CLISTAT client objects collected for Patch Manager processing.

Figure 1 Messaging Server Processing



Legend

- a** On the Configuration Server, the ZTASKEND rexx method is called at the end of the client 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, the PATCH_STATUS method in the PRIMARY.SYSTEM.ZMETHOD class of the Radia Database calls QMSG for the PATCH objects.

- b** QMSG assembles object data into XML files and creates header files with the appropriate meta data “address” needed to deliver the message by the Messaging Server. When the two message files (XML and meta data files) are created, QMSG places these files in one or more predefined data queues on the Configuration Server.
 -  Prior to Messaging Server version 3.0, QMSG placed all files in a single queue location (that is, the `..\data\default` folder). For processing efficiencies, QMSG now places objects in separate queues, based on the parameters specified in ZTASKEND and in the PATCH_STATUS method. The Messaging Server is configured to use individual Data Delivery Agents (DDAs) to process messages from these queues. Table 1 below lists the Data Delivery Agents that operate on each data queue location.
- c** The Messaging Server polls the queues and automatically picks up and transfers data files to the appropriate external locations using the routing type and locations defined in the specific data delivery agent's configuration file.

Table 1 Data Queues and Data Delivery Agents

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

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

About the Data Delivery Agents

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

- The CORE message data refer to client objects typically exchanged during a standard client connect process. Examples of CORE type message objects include ZMASTER, SESSION, and ZCONFIG.

- The INVENTORY message data is comprised of FILEPOST objects created during a client audit process.
- The WBEM message data is comprised of wbem reporting object data.
- The PATCH message data is comprised of ZOBJSTAT and CLISTAT client object data.

These Data Delivery Agents (DDAs) have the ability to post messages using ODBC into a SQL database that can be used for reporting. Using the DDAs to post messages directly into the SQL database avoids the bottlenecks created when posting multiple message types into an Inventory Server because each DDA works independently on its own queue.

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



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

The data delivery agent modules are located in the `\MessagingServer\modules` folder. The modules are loaded using “`dda.module load`” statements in the Messaging Server configuration file (`rms.cfg`).

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

About Routing Inventory Data

This Messaging Server supports the following alternative routing of information to a back-end database for the Inventory Manager:

- Direct posting via ODBC to a SQL Inventory Database – 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.)

OR

- Indirect posting of the CORE, INVENTORY, and WBEM data messages to an Integration Server using HTTP. This routing option requires the Inventory Manager Server to post the data to the back-end database.



This earlier configuration remains supported, but requires modifications to several modules after installation. For details, see *Reconfiguring the Messaging Server for RMS 2.x Processing* on page 64.

The Inventory Manager Server is discussed in the *Installation and Configuration Guide for the Inventory Manager Using Radia (Inventory Manager Guide)*.

About the SQL Database Tables and Scripts

The Data Delivery Agents for CORE, WBEM and INVENTORY post their message data into the same SQL tables created by the Inventory Manager Server. These Data Delivery Agents use the exact same table definitions used by the Inventory Manager Server to create tables, update data and delete data. If the SQL tables have not been already created by an instance of the Inventory Server, when the Data Delivery Agent that uses the SQL table is started, the table will be created. The default definitions for these tables and associated SQL queries are found in the `/etc/<module name>/sql/hp` directories of the Messaging Server. Customized scripts can be placed in the `/etc/<module name>/sql` directory; this 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`. This script is identical to the version of `taskend.tcl` on the Inventory Server. The script necessary to map the INVENTORY object data (FILEPOST object) is called `filepost.tcl`. Both of these scripts are found in the `/etc/<module name>/lib` directory of the Messaging Server. Using the identical scripts found on the Inventory Server allows previous users of this Infrastructure service to migrate any *customized* scripts directly into the `/sql` directory for the associated Data Delivery Agent module.

Version 3.1 of the Messaging Server introduces a configuration parameter to control when the SQL tables are created and the commands are loaded into memory. The default and HP-recommended behavior is to perform these SQL tasks upon the *first message delivery*. HP recommends using this setting whenever possible because it allows only the necessary commands to be loaded and is a more efficient use of resources. Alternatively, the STARTUPLOAD configuration parameter can be used when posting data using ODBC to have SQL tasks performed upon Messaging Server startup. For more information, see the STARTUPLOAD configuration parameter definition on page 97.

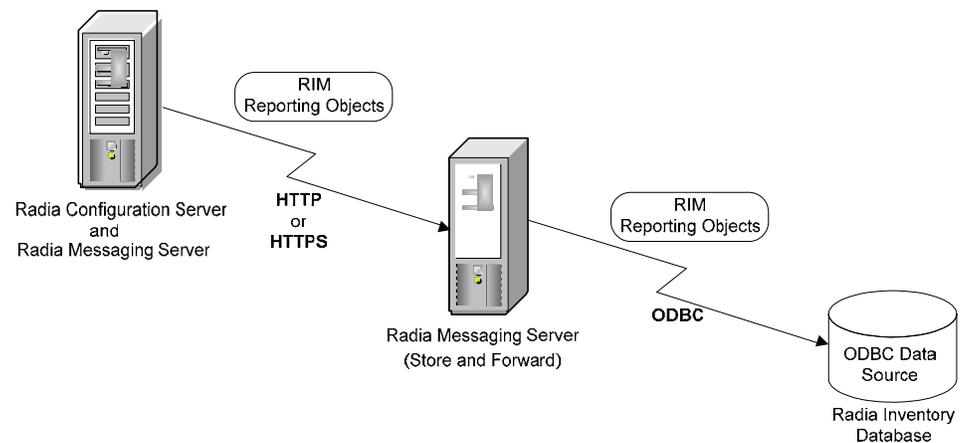
About Using Store and Forward

The Messaging Server includes store and forward capabilities that allow you to forward messages to another Messaging Server using HTTP or HTTPS. For example, a good practice before posting messages using ODBC is to forward messages to another Messaging Server located as close to the SQL database as possible.

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

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

Figure 2 Store and Forward Messaging Server



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

For more information on this topic, see Example 1: Configuring the Messaging Server for Store and Forward on page 120.

About this Guide

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

- **Installing the Messaging Server**
This chapter describes how to install the Messaging Server co-resident with the Configuration Server, and how to start and stop the Messaging Server service.
- **Configuring and Tuning the Messaging Server**
This chapter describes how the Configuration Server ZTASKEND rexx and the QMSG executable work hand-in-hand with the Messaging Server. It also discusses how to configure the Messaging Server configuration file, which loads the Data Delivery Agents. In addition this chapter also describes how to configure the DDA modules to route CORE, INVENTORY, WBEM and PATCH message data to the Inventory, Management Portal, and Patch databases or directories. Additional tuning options are included.
- **Troubleshooting**
This chapter reviews how to resolve common error messages in the `rms.log` files and identifies solutions for typical posting problems.
- **Additional Messaging Server Configuring Options**
This appendix describes alternate configurations, including how to install and configure for Store and Forward, and other customized configurations.

Summary

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

2 Installing the Messaging Server

At the end of this chapter, you will:

- Know how to install the HP OpenView Messaging Server Using Radia (Messaging Server).
- Be able to verify the installation of the Messaging Server.

Messaging Server Installation

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

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

Review the reference documentation on the HP Technical Support Web site to help you determine which machine is best suited in your environment for running the Messaging Server. Install the Messaging Server from the Extended Infrastructure directory on the Radia Infrastructure CD-ROM.

Previous releases of the Messaging Server supported the configuration of multiple worker processes operating on the same queue. This is no longer a recommended practice and will not improve the throughput of the RMS. With the introduction of the data delivery modules, multiple worker processes are no longer needed because of the parallel processing of each dda module.

Platform Coverage

The Messaging Server runs on the Windows and UNIX platforms listed in Table 2 below. These include all platforms on which the Configuration Server runs.

Table 2 Supported Operating Systems and Minimum Levels

Platform	Operating System and Minimum Level
Windows	NT 4.0 Server, Service Pack 6
	2000 Service Pack 3

Platform	Operating System and Minimum Level
	Server 2003, Service Pack 1
	XP Professional Service Pack 2
UNIX	HP-UX (PA-RISC 1.1 and 2.0), Version 10.20
	Red Hat Enterprise Linux Version 2.1.
	Red Hat Enterprise SuSE Enterprise Server, Versions 8 and 9.
	Solaris, Version 2.7
	AIX, Version 4.2

Tips

- Click **Cancel** in any of the windows to exit the installation. If you click **Cancel** accidentally, prompts enable you to return to the installation program.
- Click **Back** at any time to return to previous windows. All the information that you entered thus far will remain unchanged.
- Most windows have associated error messages. If your specifications are invalid, an error message will appear. Click **OK** and enter the correct information.
- This installation program will display recommended default values. Deviation from these default settings must be coordinated with changes in the ZTASKEND rexx. We recommend accepting all defaults for folder names and locations; however, they can be overridden by specifying the parameters necessary to suit your environment.
- The set of prompts to configure either the Messaging Server or 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 RMS install using the same installation program. Once a DDA is installed and its data-specific directory exists, the ZTASKEND method on the Configuration Server automatically redirects the messages to the new directory location.

Installation Procedures for Windows and UNIX



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

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

Overview of Installation Tasks

Because the Messaging Server installation supports prior and current configuration options, there can be many prompts for information during the install that follows. Use Table 3 on page 29 as a roadmap to the sequence and contents of the prompts.

Table 3 Messaging Server Installation -- Task Overview

Task	Page	Notes and Tips
1 Launch install, accept license and select the Messaging Server directory	30	Rename an existing <code>rms.cfg</code> file before you begin.
2 Select Data Delivery Agents to Install	31	Each DDA selected here adds the following windows: <ul style="list-style-type: none"> • Scan directory window in Step 3. • Configuration windows for Tasks 5, 6, 7 and 8. Using Data Delivery Agents is a best practice that improves performance and allows for flexible configurations and easy upgrades.
3 Identify Message Directories to Scan (Message Folder Locations)	33	You are always prompted for the Messaging Server directory to scan, followed by directories to scan for each Data Delivery Agent selected in Task 2. The Patch Directory to Scan must match the <code>-queue</code> value in <code>PRIMARY.SYSTEM.ZMETHOD.PATCH_STATUS.ZMTHPRMS</code> . See page 61 for details.
4 RMS Configuration -- For Non-DDA Message Routing Only	38	If routing objects from Data Delivery Agents, simply click through these windows. Use these windows to maintain an RMS v2.x configuration and route objects from the <code>\data\default</code> directory.
5 Configure the Core Data Delivery Agent (<code>core.dda.cfg</code>)	45	Displays if the CORE DDA was selected in Step 2. Also routes RMP objects, if desired.
6 Configure the Inventory Data Delivery Agent (<code>inventory.dda.cfg</code>)	50	Displays if INVENTORY DDA was selected in Step 2. Routes the FILEPOST objects to an Inventory database.
7 Configure the Wbem Data Delivery Agent (<code>wbem.dda.cfg</code>)	53	Displays if WBEM DDA was selected in Step 2.

Task	Page	Notes and Tips
8 Configure the Patch Data Delivery Agent (patch.dda.cfg)	56	Displays if PATCH DDA was selected in Step 2.
9 Review Installation Summary and Finish	60	Allows review before proceeding with the install.

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

1 From the Radia Infrastructure CD-ROM, navigate to the Extended_Infrastructure\MessagingServer directory. Open the folder for your operating system.

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

On a UNIX platform, enter the following command:

```
./install
```

and press **Enter**.

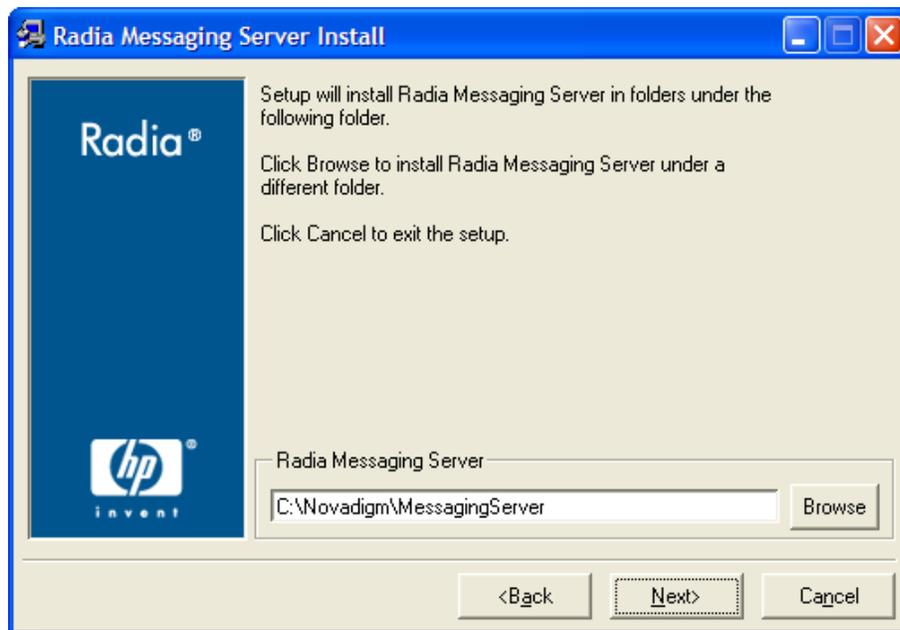
The Welcome window for the Messaging Server Setup program opens.

3 Click **Next**.

The End User License Agreement window opens.

4 Read the license agreement and click **Accept**.

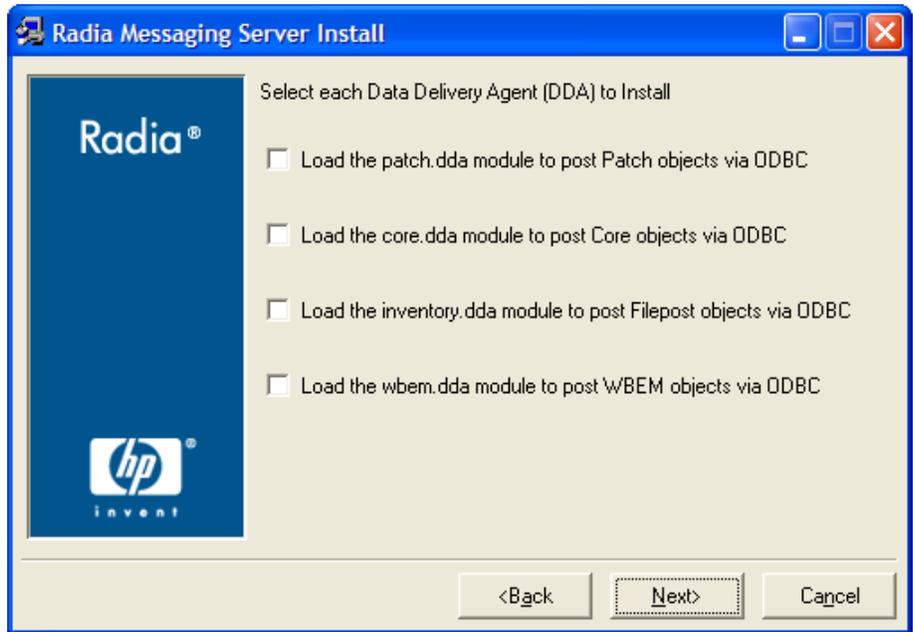
The Select the installation folder window opens.



- 5 Use this window to select the folder where you want to install the Messaging Server.
 - Click **Next** to accept the default installation folder.
 - OR
 - Click **Browse** to select a different folder.
- 6 Click **Next**.

Task 2 Select Data Delivery Agents to Install

The Select each Data Delivery Agent to install window opens.



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



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



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

Table 4 DDAs to Install by HP OpenView using Radia Product

Product	Data Delivery Agents to Install
Application Management Profiles for Server Management	core.dda
Inventory Manager	core.dda, inventory.dda and wbem.dda
Management Portal	core.dda
Patch Manager	patch.dda

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

The Default Message Directory to Scan window opens.



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

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

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

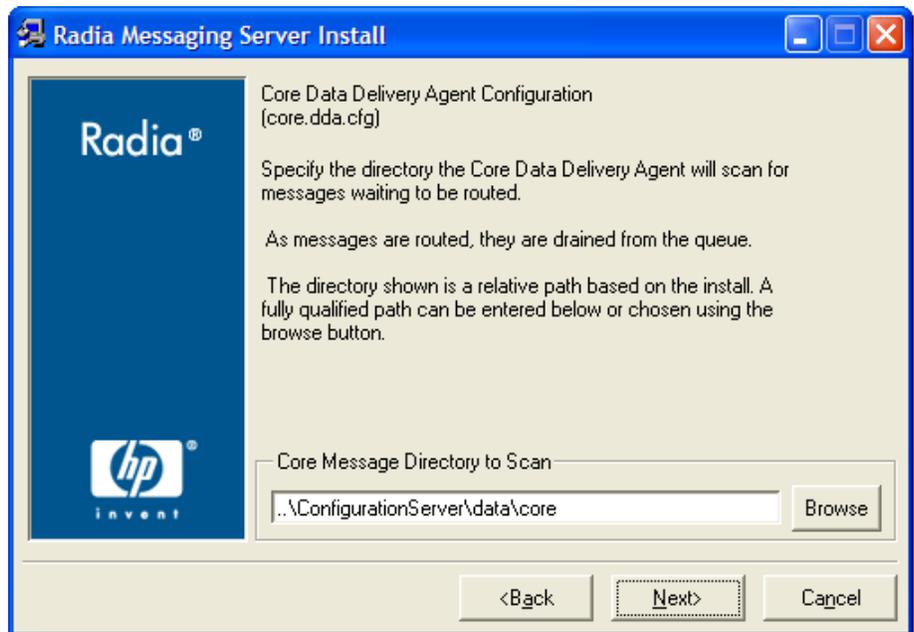


If necessary, adjust the directory path to your Configuration Server, but keep the `\data\default` folder names.

- 9 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.



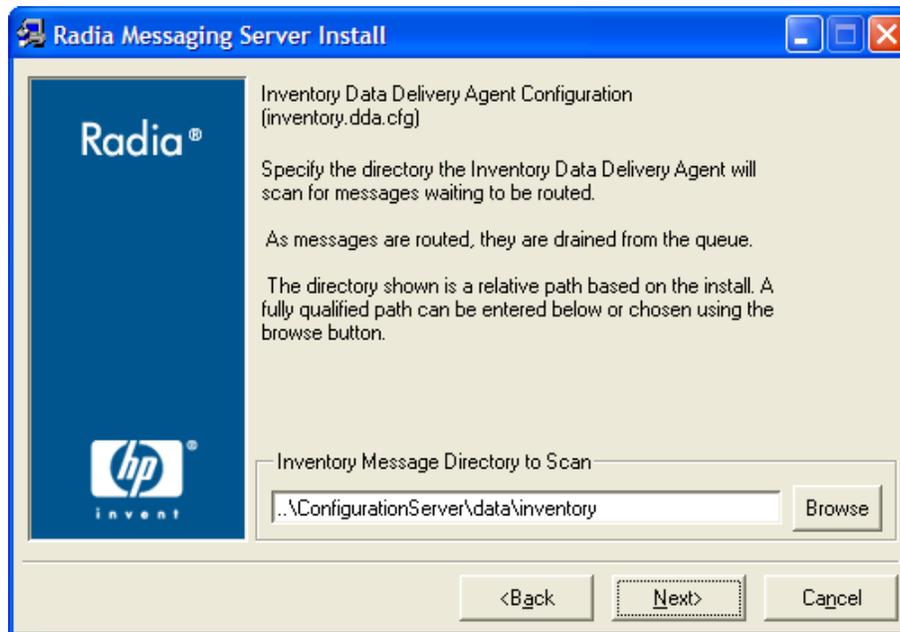
- 10 Accept the default location, or click **Browse** to select the directory where the Core Data Delivery Agent should scan for any core messages.

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

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

- 11 Click **Next**.

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



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

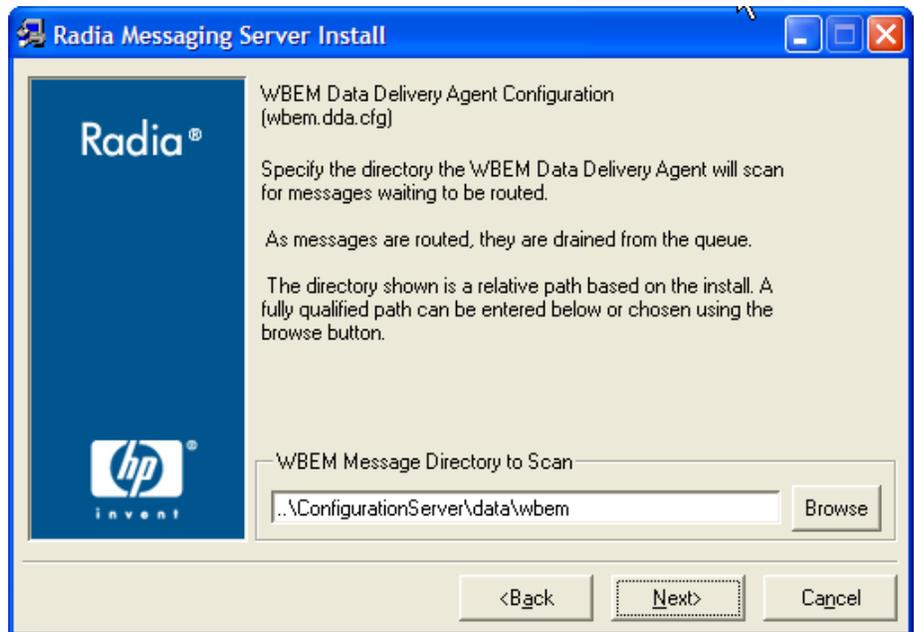
Normally, this is the `\data\inventory` folder located where the Configuration Server is installed. This directory will be created if it doesn't already exist when the Messaging Server starts. Changes to this directory name must be coordinated with changes to ZTASKEND REXX for a Messaging Server co-resident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



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

- 13 Click **Next**.

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



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

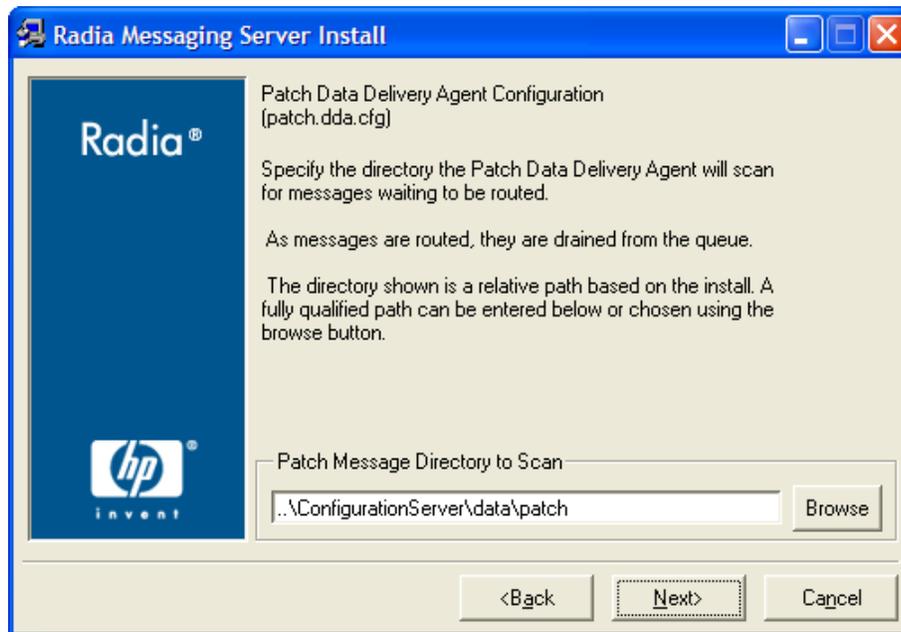
Normally, this is the `\data\inventory` folder located where the Configuration Server is installed. This directory will be created if it doesn't already exist when the Messaging Server starts. Changes to this directory name must be coordinated with changes to ZTASKEND REXX for a Messaging Server co-resident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



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

- 15 Click **Next**.

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



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

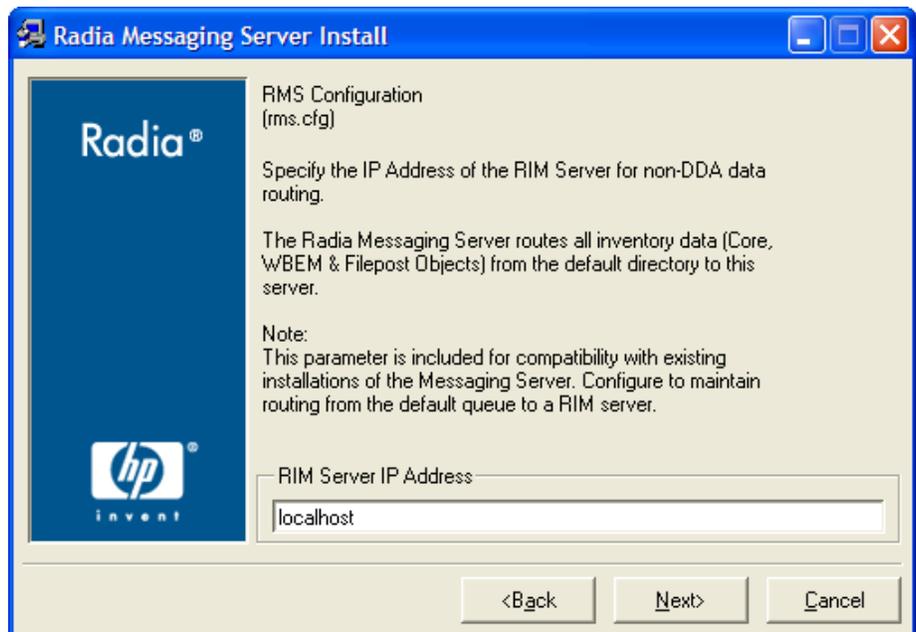
Normally, this is the `\data\patch` folder located where the Configuration Server is installed. This directory will be created if it doesn't already exist when the RMS starts up.

- If necessary, adjust the directory path to your Configuration Server. HP recommends keeping the `\data\patch` folder name.
- For a Messaging Server co-resident with the Configuration Server, if you enter a directory name other than `patch`, you must modify the `ZMTHPRMS -queue` value in the `PATCH_STATUS` method instance to match. For details, see [Verify the Patch Method Connection and Queue Name](#) on page 61.

The first RMS Configuration window opens.

Task 4 RMS Configuration – For Non-DDA Message Routing Only

- ▶ If you elected *not* to install a Data Delivery Agent to route messages of a given type, use the following set of seven installation windows to have the Messaging Server route those messages. Messaging Server routing is established by entries in the `rms.cfg` file.
- ▶ *Tip!* If you selected Data Delivery Agents to be loaded for all objects in your environment, simply click **Next** to quickly skip through all seven RMS Configuration windows (`rms.cfg`).



- 17 If you are not installing Data Delivery Agents to route all inventory objects, type the IP address or DNS host name of the Inventory Manager Server.

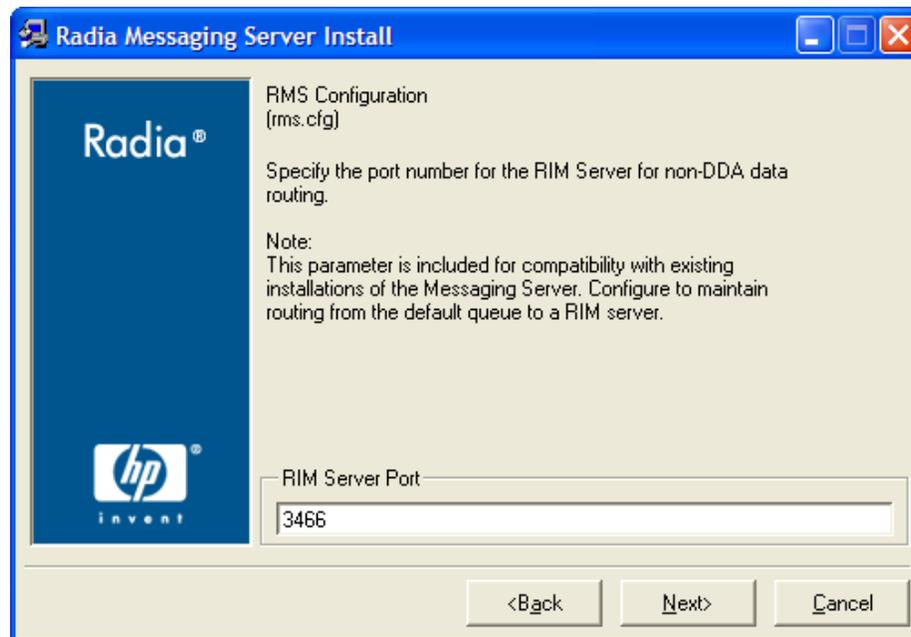
The Radia Messaging Service will route any inventory objects from the `\data\default` directory to this server using HTTP.



Following installation, you can modify `rms.cfg` file to add failover processing for an alternate Inventory Manager server. See *Configuring for Failover* on page 106 for more information.

18 Click **Next**.

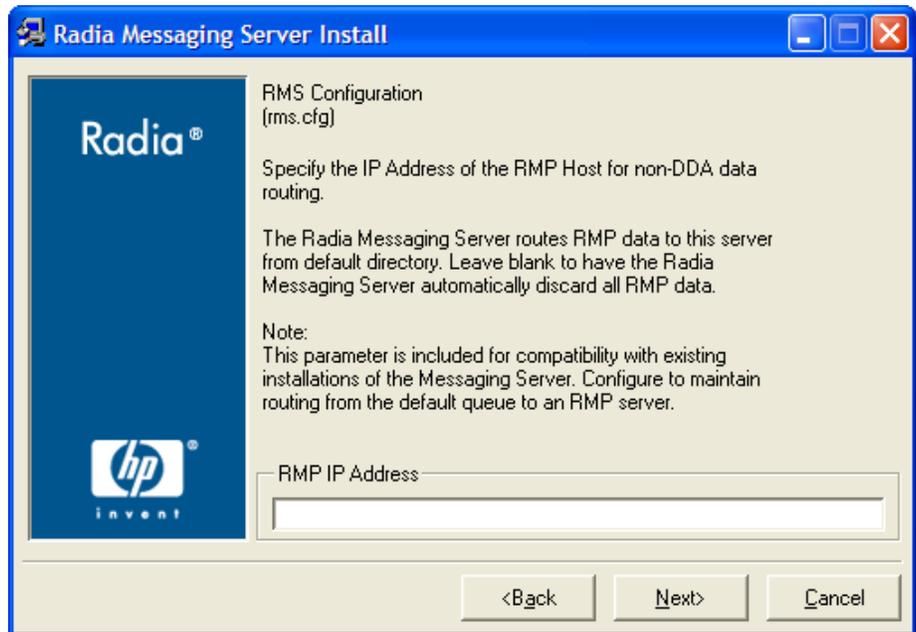
The port for the Inventory Manager Server window opens.



19 Type the port number of the Radia Inventory Server entered on the previous window. Normally, this is 3466.

20 Click **Next**.

The Settings for the Management Portal window opens.



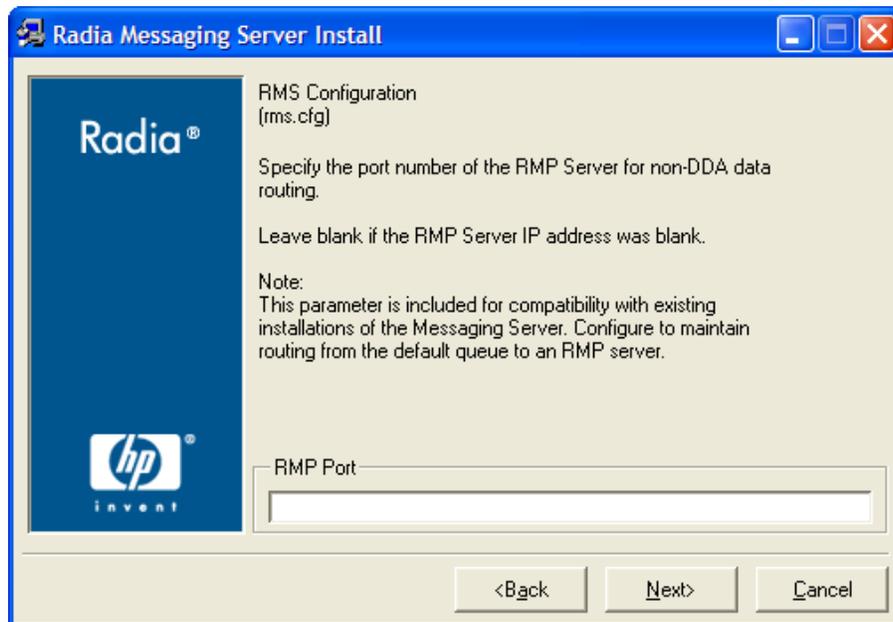
- 21 Type the IP address or DNS host name of the Management Portal server to route all RMP data found in the default scan directory location to that server.

or

To automatically discard any Management Portal data found in the default scan directory location, leave the RMP IP Address field blank.

- 22 Click **Next**.

The Management Portal Port window opens.



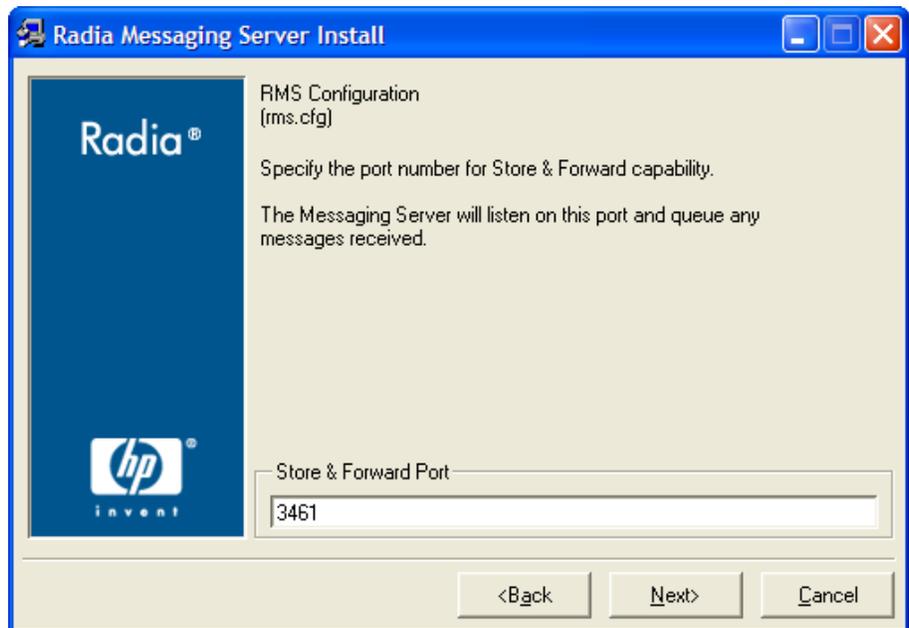
- 23 If you entered an RMP IP Address on the previous window, type the port number of the Management Portal. Normally, this is 3466.

or

Leave the RMP Portt field blank if you also left the RMP IP Address field blank.

- 24 Click **Next**.

The Store & Forward Port Setting window opens.



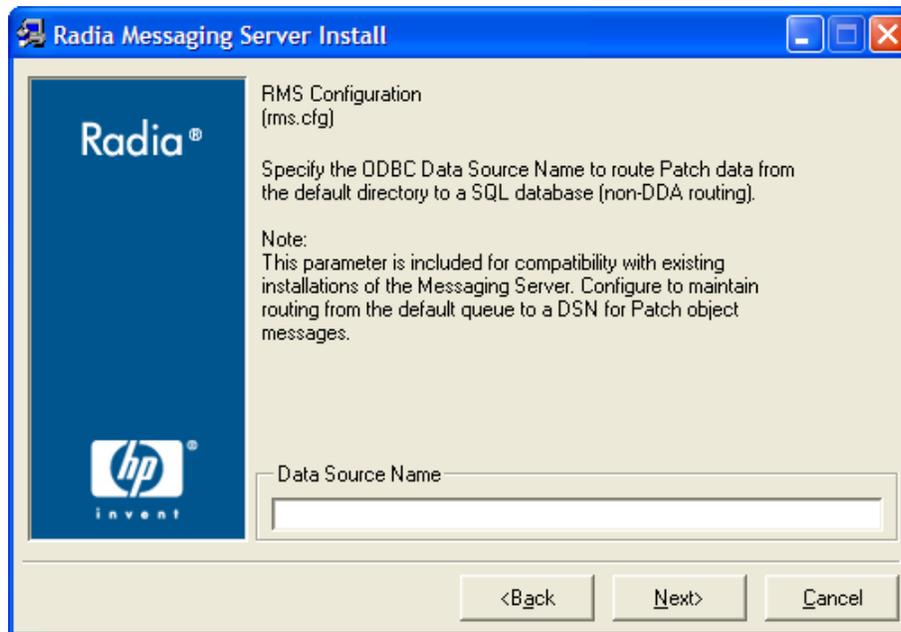
The Messaging Server includes the ability to receive and process messages that have been forwarded from other Messaging Servers in your enterprise. The port used to receive these messages is called the Store & Forward port.

For more information on using store & forward, see Example 1: Configuring the Messaging Server for Store and Forward on page 120.

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

The RMS Configuration window for the Data Source Name for ODBC routing of Patch Data opens. If the Messaging Server is to route Patch messages directly from the default scan directory, specify the ODBC settings to connect to the SQL database on the next three windows.

- ▶ Leave these three ODBC Data Set windows blank if you are installing the Patch Data Delivery Agent, or if the routing of Patch data is not needed in your environment or you can duplicate the ODBC DSN settings used to scan the default directory and then used for the patch data delivery agent module.

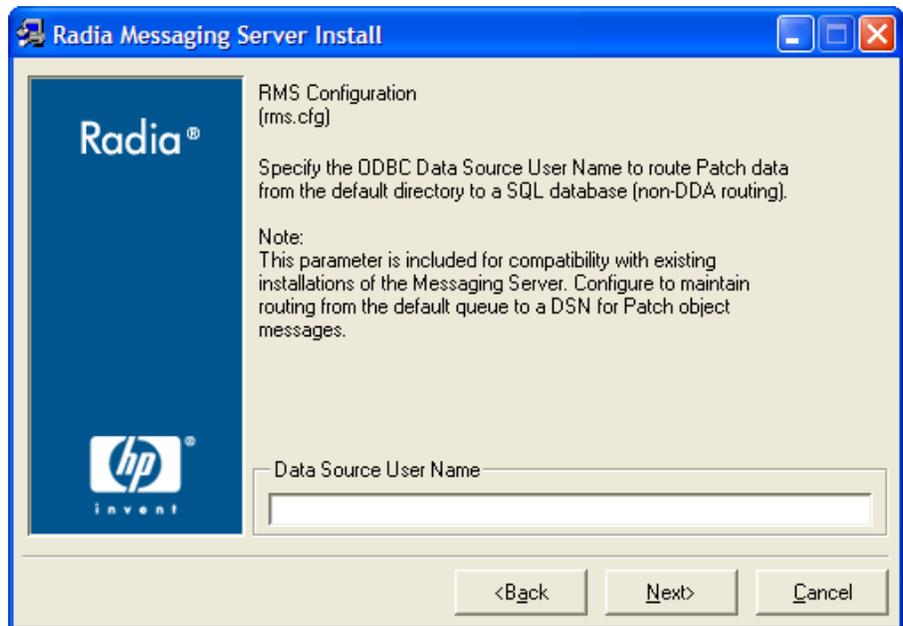


- 27 If you did not install the Patch Data Delivery Agent, type the Data Source Name of the ODBC SQL database for Patch Manager.

Leave blank if using the Patch Data Delivery Agent or if the delivery of Patch data is not required.

- 28 Click **Next**.

The RMS Configuration for the DSN User Name for routing Patch data window opens.



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

Leave blank if using the Patch Data Delivery Agent or if the delivery of Patch data is not required.

- 30 Click **Next**.

The RMS Configuration for the DSN Password for routing Patch data window opens.



- 31 Type the password required for the DSN user entered on the previous dialog.

Leave blank if using the Patch Data Delivery Agent or if the delivery of Patch data is not required.

➤ The password will be encrypted.

- 32 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`)

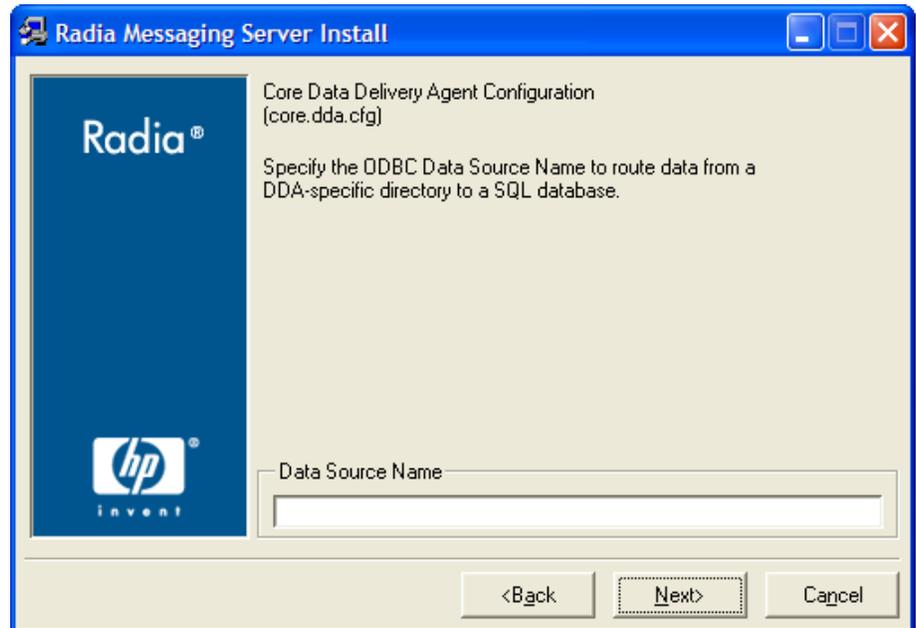
➤ If you elected to load the Core Data Delivery Agent, the five Configuration windows for the Core DDA allow you to specify:

- 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 Management 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.

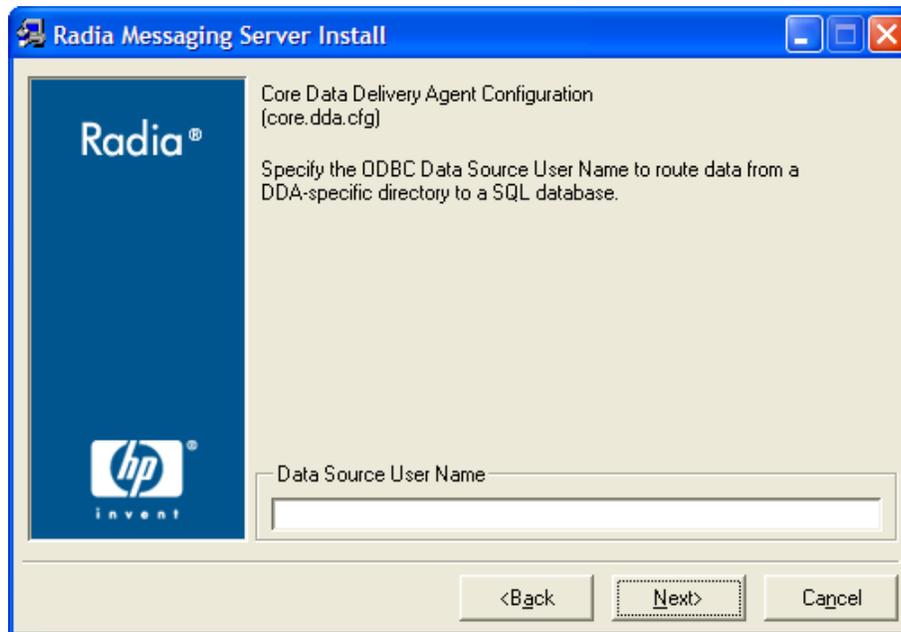
The Core Data Delivery Agent Configuration (`core.dda.cfg`) window opens for the Data Source Name for posting Core objects.



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

34 Click **Next**.

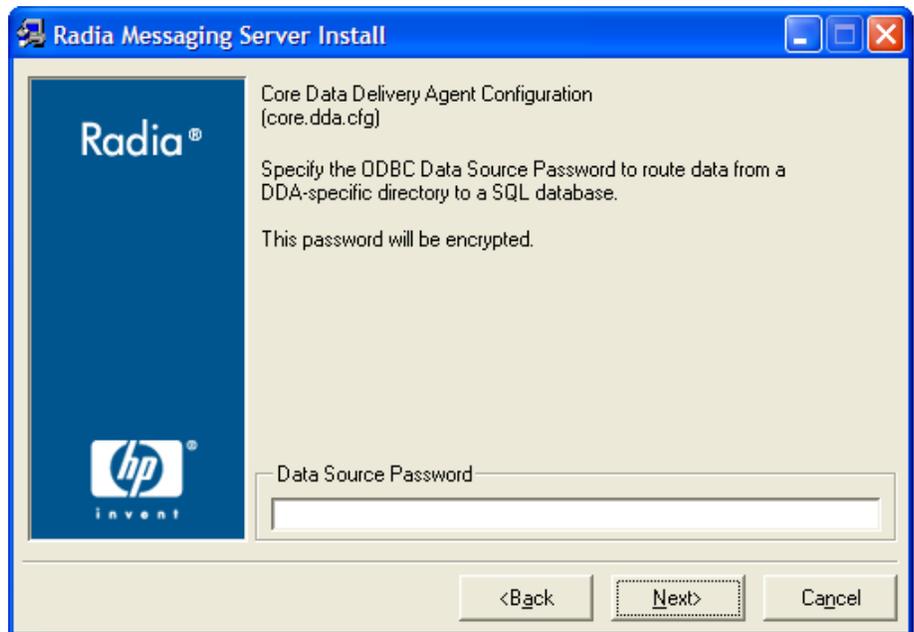
The Core Data Delivery Agent Configuration (`core.dda.cfg`) window opens for the DSN User Name for routing Core data.



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

36 Click **Next**.

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

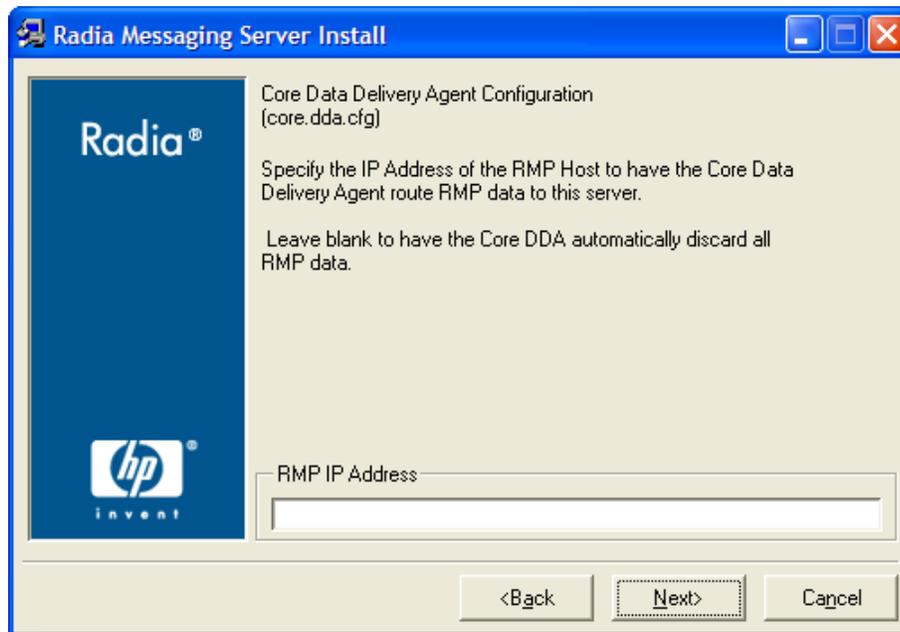


37 Type the password required for the DSN user entered on the previous window.

➤ The password will be encrypted.

38 Click **Next**.

The Core Data Delivery Agent Configuration Settings for the Management Portal window opens.



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

or

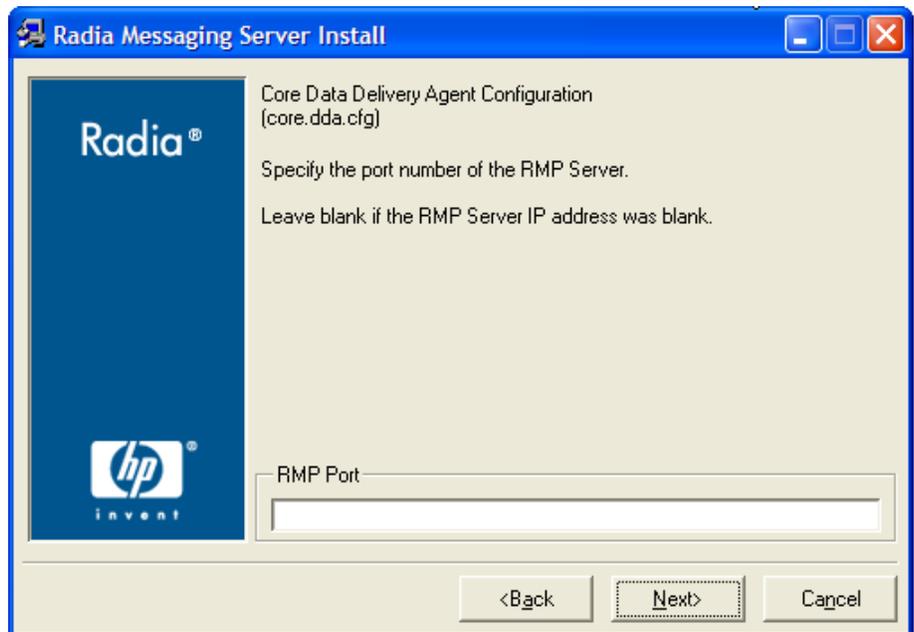
To automatically discard any Management Portal data found in the Core DDA scan directory location, leave the RMP IP Address field blank.



The Management Portal Version 2.1 (and above) no longer requires the routing of CORE.RMP data for Wake-On-Lan Notify support. This previous requirement has been removed.

- 40 Click **Next**.

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



- 41 If you entered an RMP IP Address on the previous window, type the port number of the Management Portal. Normally, this is 3466.

or

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

- 42 Click **Next**.

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

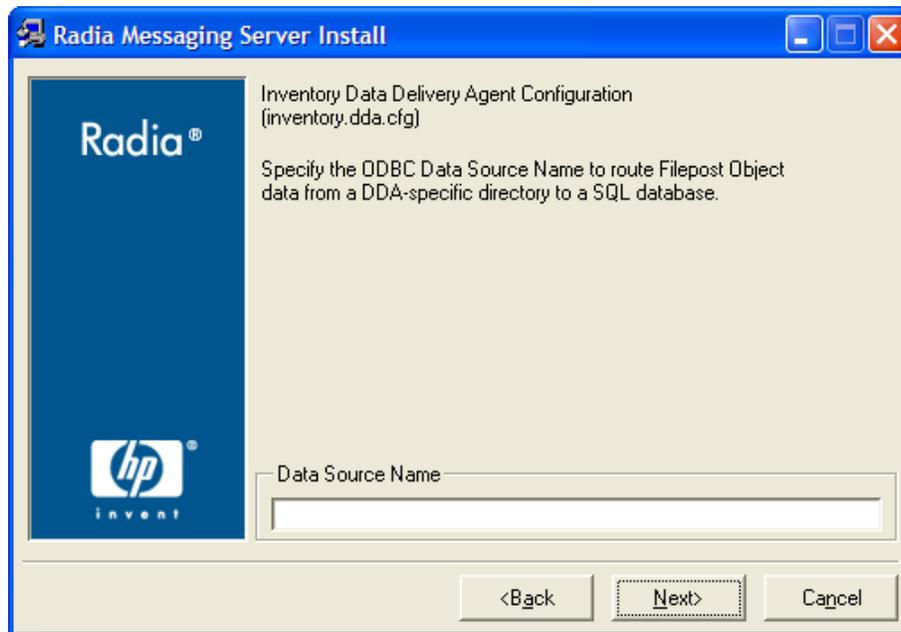


If you elected to load the Inventory Data Delivery Agent, the three Configuration windows for the Inventory DDA allow you to specify the ODBC settings to connect to the SQL Inventory database for posting Filepost objects.

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.

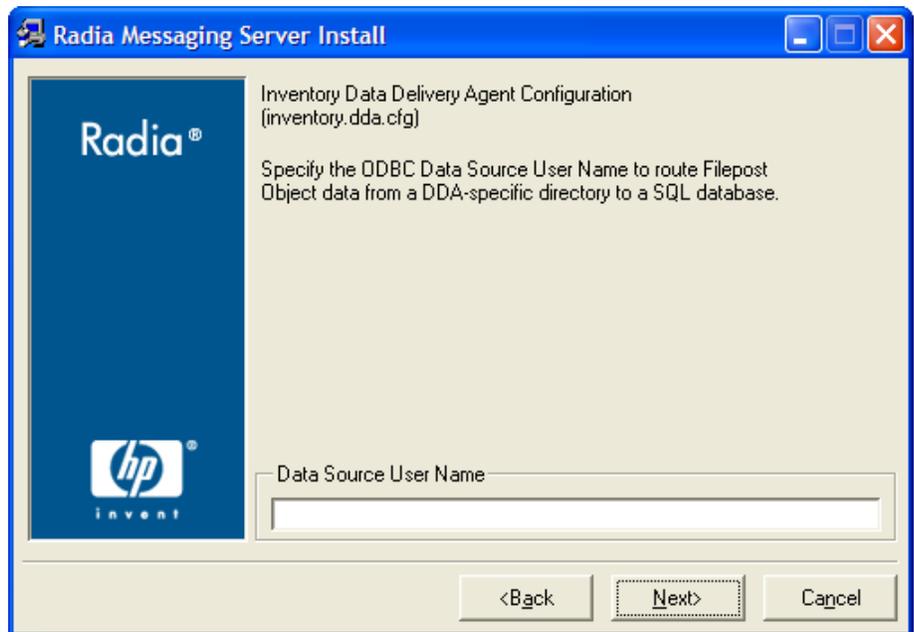
The Inventory Data Delivery Agent Configuration (`wbem.dda.cfg`) window opens for the Data Source Name for routing Inventory data.



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

44 Click **Next**.

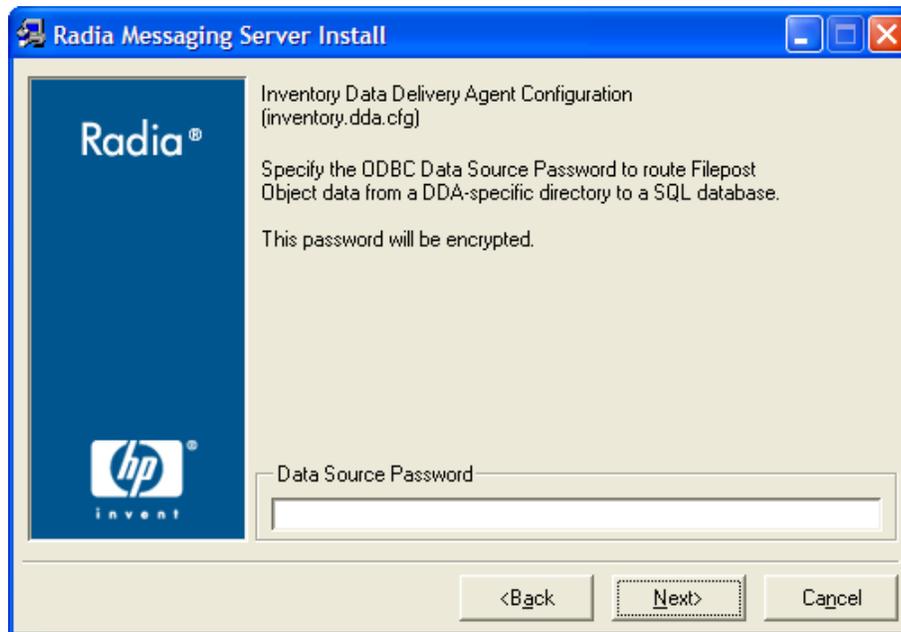
The Inventory Data Delivery Agent Configuration (`inventory.dda.cfg`) window opens for the DSN User Name for routing INVENTORY message data.



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

46 Click **Next**.

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



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

➤ The password will be encrypted.

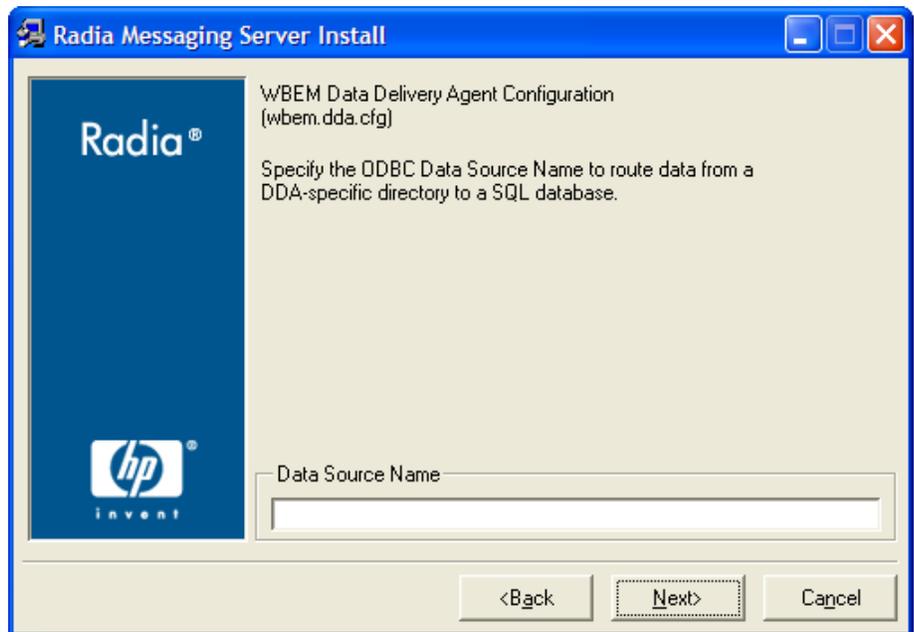
48 Click **Next**.

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.

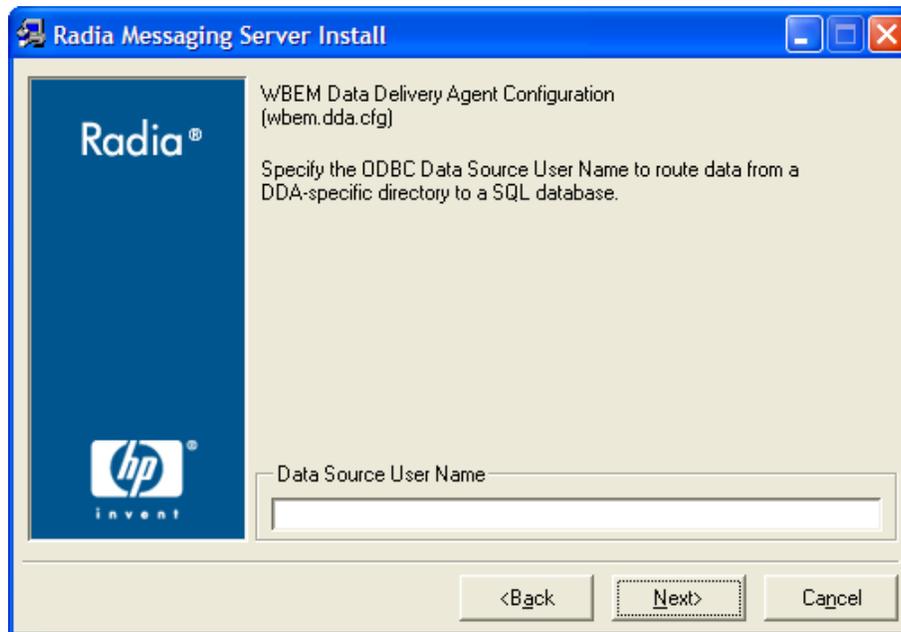
Specify the ODBC settings to connect to the SQL database for posting Wbem data in the next three windows.

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



- 49 Type the Data Source Name of the ODBC SQL database for Wbem data.
- 50 Click **Next**.

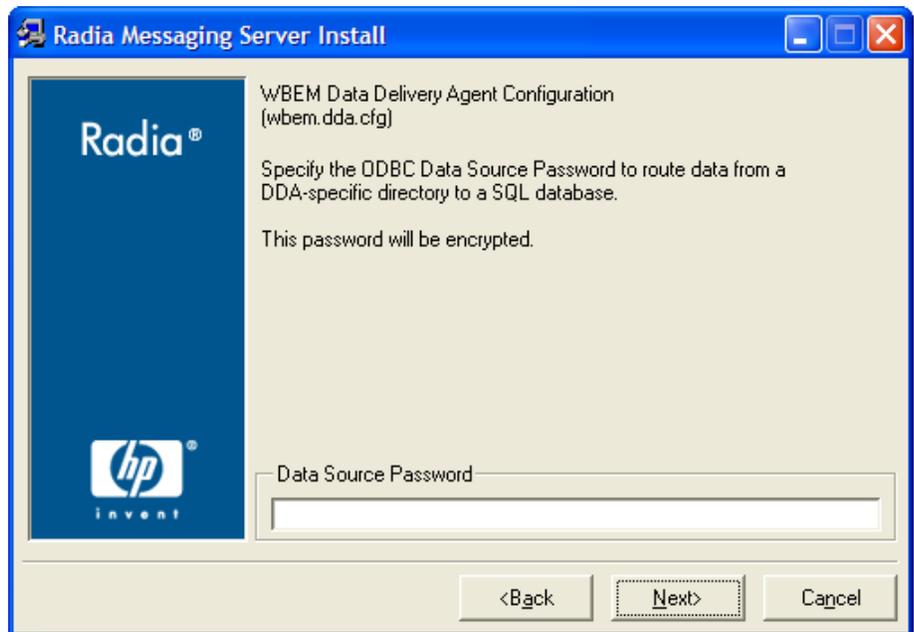
The Wbem Data Delivery Agent Configuration (`wbem.dda.cfg`) window opens for the DSN User Name for routing Wbem data.



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

52 Click **Next**.

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



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

➤ The password will be encrypted.

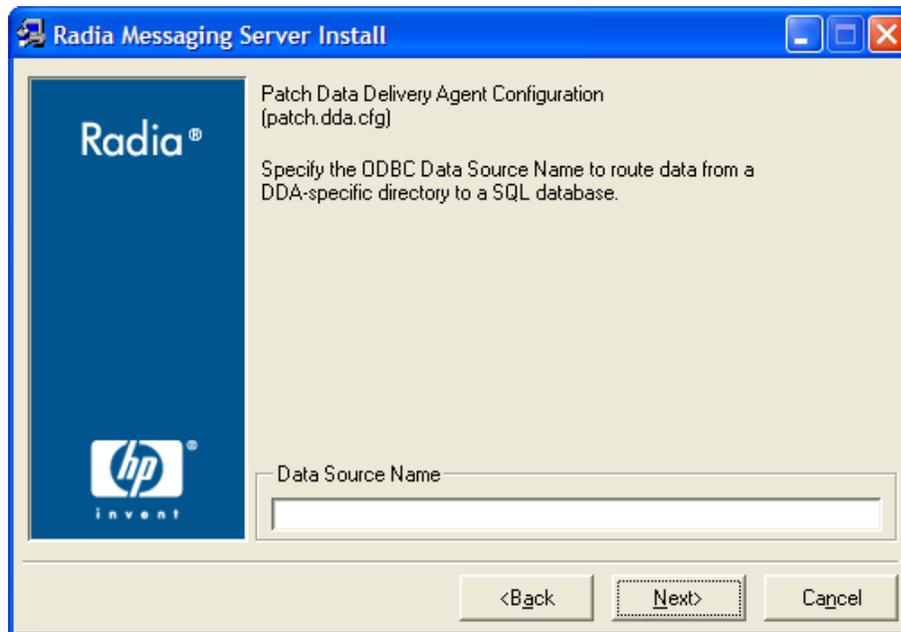
54 Click **Next**.

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.

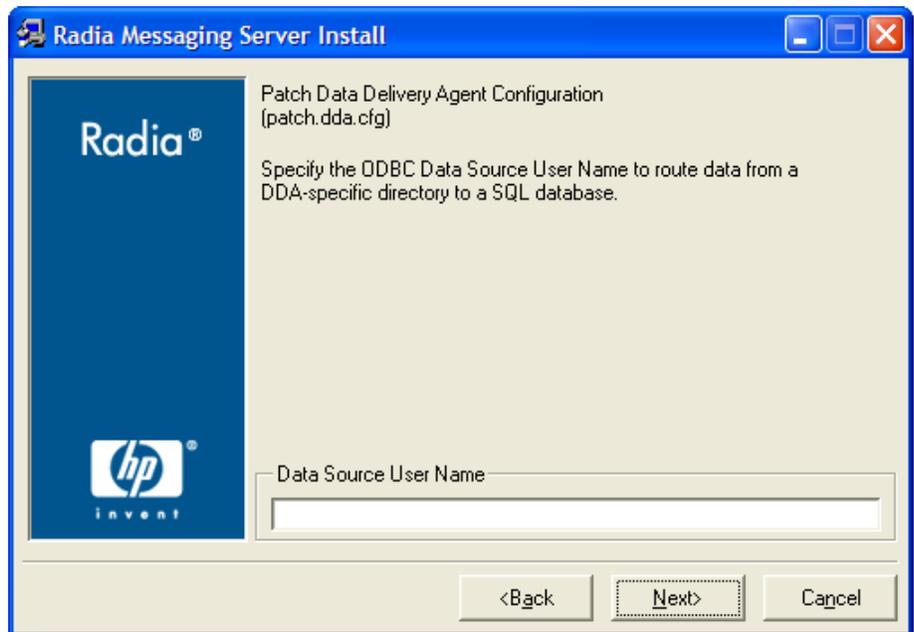
The Patch Data Delivery Agent Configuration (`patch.dda.cfg`) window opens for the Data Source Name for routing Patch data.



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

56 Click **Next**.

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



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

58 Click **Next**.

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



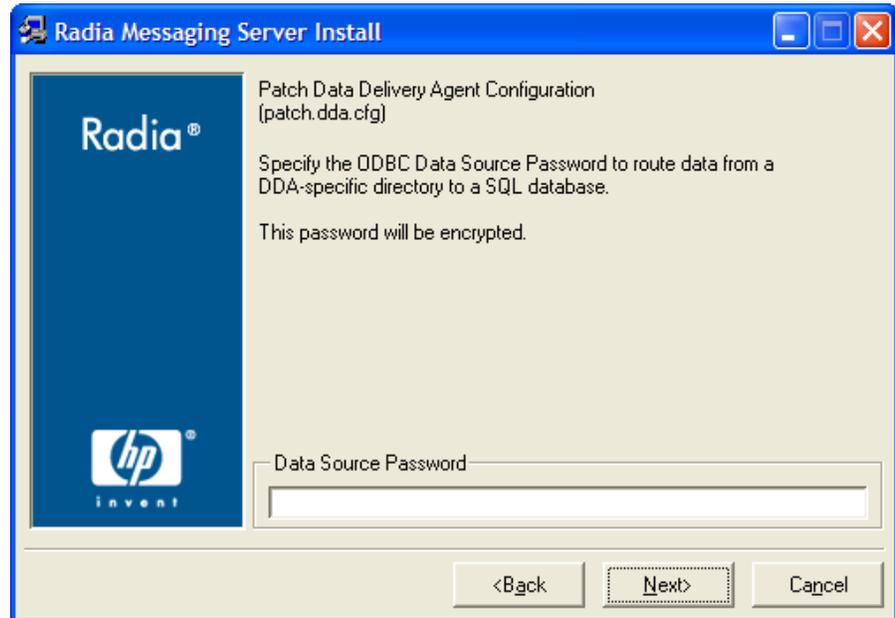
59 Type the password required for the DSN user entered on the previous window.

▶ The password will be encrypted.

60 Click **Next**.

Task 9 Review Installation Summary and Finish

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



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

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

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

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

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

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

`C:\Novadigm\MessagingServer\Logs` (on a Windows platform),

or

/opt/Novadigm/MessagingServer/Logs (on a UNIX platform).



See Additional Tuning Topics on page 106. Tuning options include changing the polling frequency or retry value, and specifying failover servers for inventory data.

Post-Installation Procedures

Use these procedures to verify the correction configurations for Patch, enable HTTPS routing using SSL, or revert to an RMS 2.x configuration.

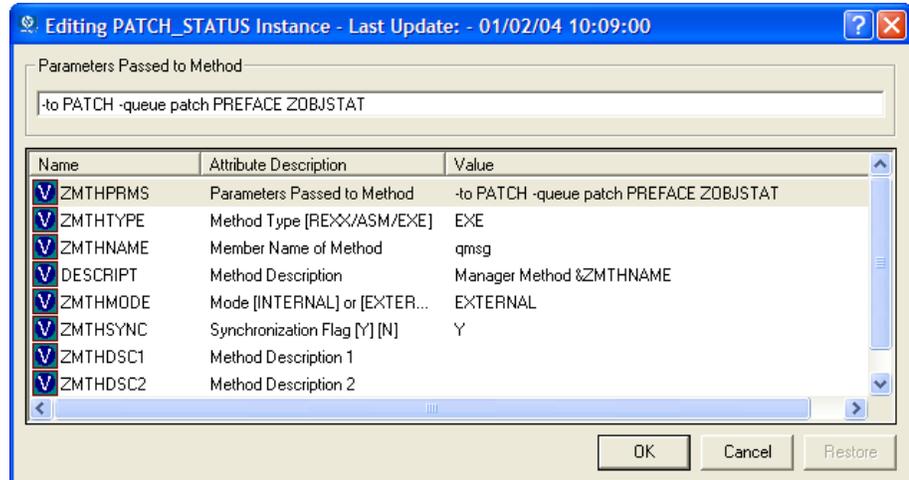
Verify the Patch Method Connection and Queue Name

- Patch Manager requires a method connection in the Radia Database. For details, refer to the *Patch Manager Guide*.
- If you installed the `patch.dda` and changed the name of the Patch Message Directory to Scan value during the Messaging Server installation (the expected value is `patch`), you must change the `-queue patch` value in the `ZMTHPRMS` attribute of the `PATCH_STATUS` instance to match the Patch Directory to Scan value.

To modify the queue name in the `PATCH_STATUS` method

- 1 Use System Explorer to edit the `ZMTHPRMS` attribute of the `PRIMARY.SYSTEM.ZMETHOD.PATCH_STATUS` instance, as shown in Figure 3 on page 62.
- 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_STATUS instance from:

```
-to PATCH -queue patch PREFACE ZOBJSTAT
```

to

```
-to PATCH -queue mypatch PREFACE ZOBJSTAT
```

- 3 Save your changes.

Enabling HTTPS Routing using SSL

The HP-OpenView Adapter for SSL Using Radia (Adapter for SSL) updates the RMS SSL Configuration Parameters in a 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.

To enable secure HTTPS routing using SSL

- 1 First update your Messaging Server to version 3.x.

- 2 Use the Adapter for SSL to install SSL support for the Messaging Server. Refer to the *Installation and Configuration Guide for the Adapter for SSL Using Radia (Adapter for SSL Guide)* for more information.

The Adapter for SSL will update the following section of the `RMS.CFG` file with the appropriate certificate and HTTPS port information:

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

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

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

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

```
msg::register rmp {  
    TYPE      HTTP  
  
    ADDRESS   {  
        PRI    10  
        URL    http://RMP_host:3466/proc/xml/obj  
    }  
}
```

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

```
msg::register rmp {
    TYPE          HTTPS

    ADDRESS      {
        PRI       10
        URL       https://RMP_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 Messaging Server service.

Reconfiguring the Messaging Server for RMS 2.x Processing

By default, this version of the Messaging Server is configured to include commands to load the Data Delivery Agent (DDA) modules. When these DDA modules are loaded, the 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 Inventory Manager Server using HTTP (which then posts them to the SQL database), two post-installation changes are required:

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

For more information, see Processing under RMS Version 2.x and RMS Version 3.x on page 80, and Example 2: Configuring the Messaging Server to Route Objects from a Single \Data\Default Queue on page 129.

Starting and Stopping the Messaging Server

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

- See the Windows procedures below.
- See the UNIX procedures below.

Windows Procedures

The Messaging Server is automatically installed as a Windows service. The service name is Radia Messaging Server (rms).

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

```
nvdkit rms.tkd start
```

```
nvdkit rms.tkd stop
```

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

```
nvdkit rms.tkd
```

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

UNIX Procedures

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

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

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

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



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

- To obtain the PID for the Radia Messaging Service, check the `rms.log` in the `\logs` directory. The PID is listed with the entry: Radia Messaging Service started (PID: XXX).

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 Messaging Server.

```
kill -9 <PID>
```

Verify Installation

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

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

- Look for the entry:

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

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

```
Info: -----  
Info: Radia Messaging Service (Version 3.0.1 - Build 60)  
Info: Radia Messaging Service - main worker - started (PID: 1180)  
Info: Platform: Windows_NT  
Info: -----
```

- 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.dca - Version 3.0.1 - Build 10.  
Info: -----
```

- 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 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 Messaging Server is installed on a Windows platform, check in the Task Manager for the `nvdkit.exe` process. If installed on a UNIX platform, check for the `nvdkit` processes running on UNIX.

The Messaging Server will start a single main `nvdkit` process and a separate `nvdkit` process for each worker process. Each DDA module installed will be a separate worker process and the Messaging Server

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

Summary

- Understand your network topology and have the targets for the Messaging Server routes laid out before installing the Messaging Server.
- To create a Messaging Server environment on your Configuration Server, the Configuration Server must include a version of ZTASKEND REXX method that calls the QMSG executable for Inventory and Management Portal data objects. To route Patch objects, the Configuration Server must have a method connection to `SYSTEM.ZMETHOD.PATCH_STATUS`.
- The Messaging Server is installed as a Windows service or a UNIX process.
- Following installation of the Messaging Server, if you opt to route data directly to an Inventory SQL database, you may be able to remove the Inventory Manager server from your environment. The Messaging Server uses the same SQL tables and queries as the Inventory Server to create, update, and maintain the Inventory SQL database. Customized SQL scripts can be ported from the Inventory Server directories to the Messaging Server directories.
- Verify installation by checking that the Messaging Service is running, it is loading the selected Data Delivery Agent modules, and there is a worker process started for both the Messaging Server and each DDA that was loaded.

3 Configuring and Tuning the Messaging Server

At the end of this chapter, you will:

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



- The first part of this chapter discusses the Configuration Server modules that support the Messaging Server.
- The topics on configuring and tuning the Messaging Server and Data Delivery Agents begin on page 84.
- To configure a Messaging Server for Store and Forward, see Example 1: Configuring the Messaging Server for Store and Forward on page 120.
- To configure a Messaging Server Data Delivery Agent to post messages to multiple DSNs, see Example 4: Configuring Data Delivery Agents to Route to Multiple DSNs using ODBC on page 136.

Understanding the Configuration Server Modules that Support the Messaging Server

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

Getting Client information to the Messaging Server (RMS)

Client objects exchanged with or created on the RCS during a client connect session are formatted into messages for the Messaging Server via the RCS binary executable QMSG. The QMSG executable is part of the standard RCS release. This executable is invoked during client taskend processing by the rexx method ZTASKEND or a connection to the method PATCH_STATUS. The calls to QMSG can include parameters specifying what queue to place the messages in, the priority in which the message is to be processed, the objects that are to be included in the messages and the “destination address” or routing identifier for the file.

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

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

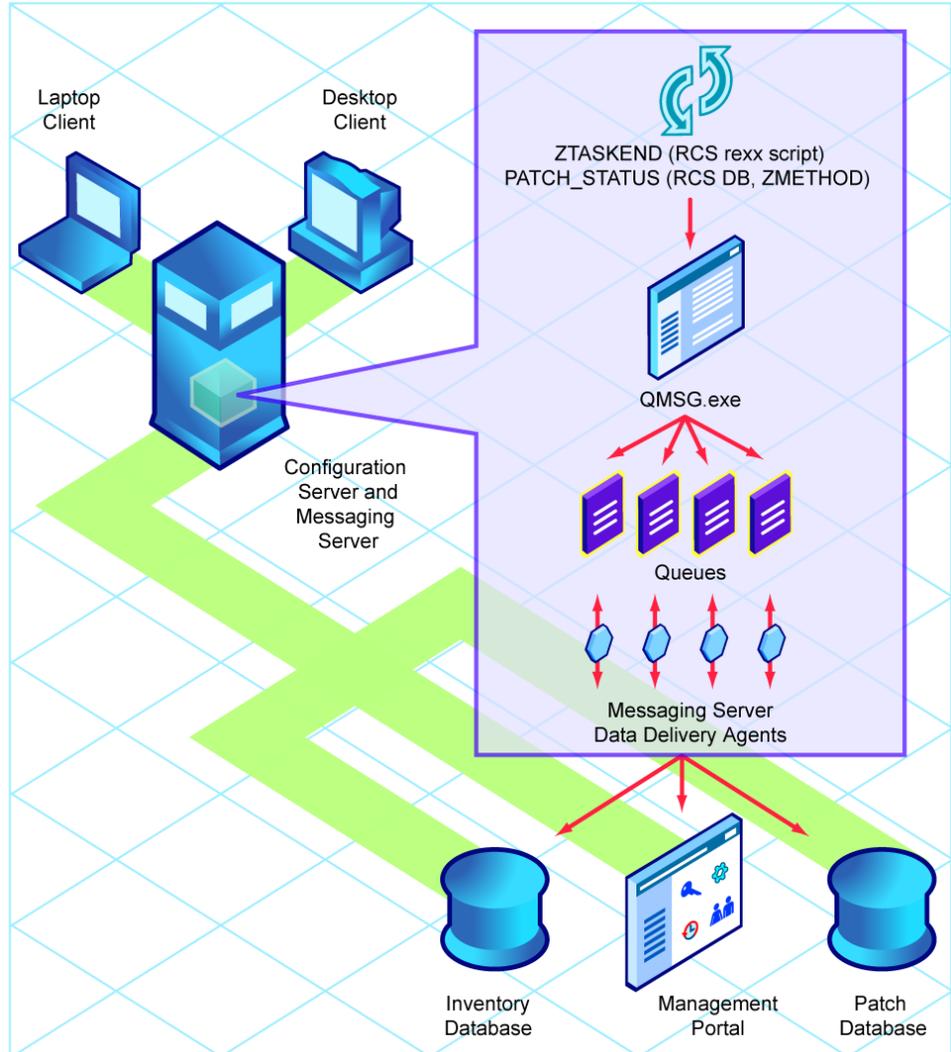
In Summary:

- For client information needed by the Patch Manager, QMSG is executed as a result of the `SYSTEM.ZMETHOD.PATCH_STATUS` method instance referenced through a method connection in the `SYSTEM.PROCESS.ZOBJSTAT` instance.

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

- For information needed by the Inventory Manager and Management Portal, the RCS 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.

Figure 4 Getting client information to the Messaging Server



About the Patch Method for Collection

The `PATCH_STATUS` method calls the `QMSG` executable with the parameters in the `ZMTHPRMS` attribute of the method. The default value of this attribute looks like this:

```
ZMTHPRMS      -to PATCH -queue patch PREFACE ZOBJSTAT
```

The `-to PATCH` 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 parameters `PREFACE ZOBJSTAT` specify the objects that will be included in the message files created by `QMSG`.

About the ZTASKEND REXX method

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

▶ As of `ZTASKEND v 1.9`, different object types (core, inventory and `wbem` objects) are placed in separate queues whenever the Data Delivery Agents have been installed for those data objects. Previous versions of `ZTASKEND` placed all data objects in a single queue (named `/data/default`).

An important job of `ZTASKEND` is to ensure unique client data is collected at the appropriate client 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 client connect phases and which objects are collected.
- The basic syntax of calls to `QMSG`.

- How the QMSG -to parameter establishes one or more destinations for message processing.
- How the QMSG -priority parameter establishes Messaging Server processing order.

ZTASKEND calls to QMSG



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

Processing Phase-Dependent Objects

Each time a Radia client connects to the RCS, the client 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 client 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)
- CLIENT SELF MAINTENANCE
- CATALOG RESOLUTION
- SERVICE RESOLUTION
- CLIENT REPORTING

Processing CORE Objects by Phase

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

For RIM: APPEVENT MSIEVENT SYNOPSIS RNPEVENT

For RMP: SYNOPSIS

In addition to the critical objects, each phase contains additional objects collected for that phase.

Table 5 Critical Objects collected by phase

Phase	Critical Objects
BOOTSTRAP (COP)	SESSION PREFACE ZSTATUS SMINFO

Phase	Critical Objects
CLIENT SELFMAINTENANCE	SESSION PREFACE ZSTATUS SMINFO
CATALOG RESOLUTION	SESSION PREFACE ZSTATUS ZCONFIG ZMASTER SMINFO
SERVICE RESOLUTION	SESSION PREFACE ZSTATUS SMINFO
CLIENT_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 a client connects during the phases CATALOG_RESO and CLIENT_REPORTING.
 - 1 If a critical object is not found, then code does not invoke QMSG for the following client phases: BOOTSTRAP (COP), SERVICE RESOLUTION and CLIENT_SELF MAINTAINANCE.
 - 2 Finally, ZTASKEND does not invoke QMSG to obtain error message objects (ZERRORM & ZERROR).

Table 6 on page 78 summarizes the Client Connect phases and the objects collected during the ZTASKEND calls to QMSG, for CORE data going to RIM or RMP.

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

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

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

Adding Items to a Critical Object List

The ZTASKEND rexx code can be configured to add object names to the critical and additional object lists. The rexx variables `CriticalRIMObjects` and `CriticalRMPObjets` 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"
CriticalRMPObjets = "SYNOPSIS"

```

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

Processing Always Objects

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

addition to the larger inventory 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 RMS Version 2.x and RMS Version 3.x

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

Since RMS version 2.x posts all objects to one queue, RMS 2.x uses the `-priority` switch of the `QMSG` call syntax to defer processing of larger inventory objects. With RMS version 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 RCS objects to RMS for RIM and RMP. `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 RMS 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 RMS version 3.x destination values used by the delivered QMSG include:

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

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

Table 7 QMSG Destinations and DDA Configuration Locations in RMS v3.x

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 RMS 2.x processing, each message destination requires an equivalent ROUTE defined for it in the msg::register router section of the rms.cfg file, as discussed in Example 2: Configuring the Messaging Server to Route Objects from a Single \Data\Default Queue on page 129.

-queue <queue>

The queue is a directory relative to where QMSG.EXE exists. QMSG is located in

the `\bin` directory of the RCS. For example, on a Windows platform, if QMSG resides at

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

Then the queues would reside at:

```
C:\Radia\ConfigurationServer\data\blue
```

```
C:\Radia\ConfigurationServer\data\default
```

```
C:\Radia\ConfigurationServer\data\green
```

```
C:\Radia\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 v. 1.9.

For Patch routing, the queue location is named in the parameters passed to QMSG from the `SYSTEM.ZMETHOD.PATCH_STATUS` method instance. For example:

```
Method = qmsg
```

```
Parameter = -to PATCH -queue patch PREFACE ZOBJSTAT
```

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

-priority

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

object1 [objectn]

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

How Priority Establishes Messaging Server Processing Order

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

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

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

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

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

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

Configuring the Messaging Server

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

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

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

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

All the configuration files for the Messaging Server and the associated Data Delivery Agents have similar configuration sections. Understanding these sections and the syntax used to configure them will aid in customizing your environment

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

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

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

A unique identifier follows the `msg::register` command. Within an instance of the RMS, which includes the configuration files for the RMS 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`)
- PATCHODBC (only configurable in `patch.dda.cfg`)
- PATCHFILTER (only configurable in `patch.dda.cfg`)

The following table gives a description of the different TYPES and their configurable parameters. The sections are listed in the general order in which they appear in the configuration files.

Table 8 Glossary of Section TYPES and Configurable Parameters

Section TYPE	Configurable Parameters
<p>QUEUE Defines the directory where messages are placed for processing.</p>	<p>DIR The directory name of the queue.</p> <p>USE Specify the name of the unique identifier that will signify how to dispatch the message. Usually this is a ROUTER type.</p> <p>POLL By default, the Messaging Server is configured to poll the queue location every 10 seconds and post up to 100 objects at a time. To change the poll interval, modify the POLL parameter.</p> <p>COUNT Maximum number of objects to post at a time (during the polling interval defined by POLL). Default is 100 objects.</p> <p>ATTEMPTS Maximum number of attempts to retry a failed message delivery before discarding the message.</p>

Section TYPE	Configurable Parameters
	<p>Default is 200 attempts. (By default, the Messaging Server and Data Delivery Agents are configured to retry any failed posts every hour, and make up to 200 attempts.)</p> <p>DELAY Number of seconds to wait between attempts to retry a failed message delivery. Default is 3600 seconds, or one hour.</p> <p>Note: To calculate the maximum time that a message could stay in the queue before being discarded, take the DELAY time and multiply it by the ATTEMPTS value. Using the default settings, this is a DELAY of 3600 seconds x 200 ATTEMPTS, or approximately eight days.</p>
<p>ROUTER Defines where the messages are going to sent.</p>	<p>ROUTE Delimit each Route by curly braces</p> <p>TO This is the address of the message. It is contained in the meta data file of each message when the message is created.</p> <p>USE Specify the unique name of a Messaging Server TYPE that will be used to dispatch the message, such as HTTP, HTTPS or ODBC. (In a DDA file, the USE entries may also be COREODBC, WBEMODBC and PATCHODBC).</p>
<p>HTTP This is a way to post the message to another server location using http protocol. The target server can be another Messaging Server where the message can be re-queued or it can be another part of the Infrastructure such as an Inventory Server or the Management Portal.</p>	<p>ADDRESS Delimit each address using curly braces. Multiple URL destinations can be specified within each HTTP TYPE as long as each has its own ADDRESS label and a different priority.</p> <p>PRI Denotes the priority in which to sent messages to the companion URL. The default priority is 10. The priority setting only matters if multiple ADDRESS entries are configured. If multiple ADDRESS entries are configured the lowest priority is tried first and if that fails the next priority URL is tried. This allows failover capability if there are network problems.</p>

Section TYPE	Configurable Parameters
	<p>URL Specifies the URL to use to send the message.</p>
<p>HTTPS This is a way to post the message using a secure socket. The HTTPS parameters are configured the same as the HTTP parameters--with the exception of the URL specification. The URL uses the https:// convention. Note: First run the HP-OpenView Adapter for SSL Using Radia to fill in the required SSL parameters in the rms.cfg file.</p>	<p>ADDRESS Same as TYPE of HTTP.</p> <p>PRI Same as TYPE of HTTP.</p> <p>URL Specifies the URL using the https:// convention.</p>
<p>HTTPD Defines the parameters the Messaging Server will use to receive incoming messages.</p>	<p>PORT Defines the Port used to “listen” for messages. Only one port specification can be used for a given Messaging Server.</p> <p>URLHANDLER Delimit with curly braces. If the incoming messages are to be deposited into different queues, depending upon the URL, the URLHANDLER must be used to delimit the USE and URL parameters.</p> <p>USE Specify the name of the QUEUE type that will receive the incoming messages.</p> <p>URL Specify the URL prefix that that will be accepted by the Messaging Server. When messages are received with the designated URL they are deposited in the associated queue. The QUEUE type must be defined when messages are received. All URL’s specified must start with /proc/.</p>
<p>ODBC Used to post PATCH messages into a SQL database.</p>	<p>DSN Data Source Name</p> <p>USER</p>

Section TYPE	Configurable Parameters
	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 <code>core.dda.cfg</code> and <code>inventory.dda.cfg</code> .	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 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 Determines when SQL tables are created and SQL scripts are loaded into memory. Default is 0, which performs these SQL tasks when the first message is posted. HP recommends using this setting whenever possible because it allows only the necessary commands to be loaded and is a more efficient use of resources. Set STARTUPLOAD to 1 to have the SQL tasks performed upon Messaging Server and Data Delivery

Section TYPE	Configurable Parameters
	Agent startup; use this setting when it is necessary to create the SQL tables upon startup due to limitations set by a Database Administrator.
<p>WBEMODBC Used to post WBEM data messages into a SQL database. Only configurable in <code>wbem.dda.cfg</code>.</p>	<p>See COREODBC for common parameters.</p> <p>AUTOCREATE A switch to enable the creation of a new SQL file and table in the Inventory ODBC database when a new object class is received. Default is 0.</p> <p>0 – Does not create a SQL file or table entry for a new object class.</p> <p>1 – Creates a new SQL file and table entry for a new object class.</p>
<p>PATCHODBC Used to post PATCH data messages into an SQL database. Only configurable in <code>patch.dda.cfg</code></p>	<p>DSN Data Source Name</p> <p>USER User ID for the DSN</p> <p>PASS Password for the DSN</p>
<p>PATCHFILTER A means to route PATCH data into multiple SQL tables. Only configurable in <code>patch.dda.cfg</code>.</p>	<p>USE Specify the name of the unique identifier that will signify how to dispatch the message.</p> <p>TO This is the address of the message. It is contained in the meta data file of each message when the message is created.</p> <p>FILTER Used for PATCH object processing into multiple tables</p>

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

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

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

- About the Sections in the RMS.CFG File below
- About the Sections in the CORE.DDA.CFG File on page 93
- About the Sections in the CORE.DDA.CFG File on page 93
- About the Sections in the WBEM.DDA.CFG File on page 100
- About the Sections in the PATCH.DDA.CFG File on page 103
- Additional Tuning Topics on page 106

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

- 1 Stop the Messaging Server before editing the `rms.cfg` file. For details, see [Starting and Stopping the Messaging Server](#) on page 65.
- 2 Edit the appropriate `*.cfg` file using any text editor. By default, the `*.cfg` files are located at: `SystemDrive:Novadigm\MessagingServer\etc` for Windows, or `/opt/Novadigm/MessagingServer/etc` for UNIX.
- 3 Modify the sections using the information given in the following topics.
 -  All path entries in the configuration files must be specified using forward slashes. This applies to both Windows and UNIX environments.
- 4 Save your changes and restart the Messaging Server.

About the Sections in the RMS.CFG File

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

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

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

Additional Sections in the RMS.CFG File

- **Required packages**
Do not remove the following lines at the top of the `rms.cfg` file

```
package require nvd.msg
package require nvd.httpd
```
- **SSL Certificates**
If the Messaging Server is SSL enabled the following sections exist in the `rms.cfg`. An `Overrides Config { }` section defines the necessary certificates and parameters for SSL support as well as the command `module load tls` which loads the code necessary to support SSL. For more information, refer to the *Installation and Configuration Guide for the HP OpenView Adapter for SSL Using Radia*.
- **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 107. The log files are located in the Logs directory of where the Messaging Server was installed.
- **log.configure -stderr 0**
Required by the 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 database and, optionally, CORE message data to the Management Portal directory. See About the Sections in the CORE.DDA.CFG File on page 93 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 database. See About the Sections in the INVENTORY.DDA.CFG File on page 98 for more information on how to configure the posting of core objects.
- **dda.module load wbem**
Posts wbem objects to a SQL database. See About the Sections in the WBEM.DDA.CFG File on page 100 for more information on how to configure the posting of wbem objects.
- **dda.module load patch**
Post PATCH message data to a SQL database. See About the Sections in the PATCH.DDA.CFG File on page 103 for information on how to configure the posting of patch objects.
- (Optional) **msg::register default, msg::register router, and msg::register <rim|rmp|other>**
These sections, if they exist, define how the Messaging Server handles the messages placed by QMSG in an existing `/data/default` location (or `/data/default queue`). For more information, see Example 2: Configuring the Messaging Server to Route Objects from a Single `\Data\Default Queue` on page 129.
 - ▶ 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 Configuring the Log Size and Number for more information. See for Configuring the Log Level, Log Size and Number on page 107 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 Client, available during the client connect process and used in reports. Examples of core objects are ZMASTER, ZCONFIG, and SESSION.

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

Queue folder for core messages: `< RCS folder> \data\core`

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

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

Several configurations are possible.

- Core data can be forward to another messaging server. This option is used to place the objects as close to the SQL database as possible before ODBC posting to avoid slow network response.
- 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 object data can be routed using HTTP to an Inventory Server. From there, the Inventory Server can post the messages to the back-end database. This option was the previous implementation method in RMS 2.x but has been replaced with the direct posting via the data delivery agents into the SQL database.
- Core messages for RMP can be routed using HTTP to a Portal Zone, or discarded using the built-in `/dev/null` location.

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

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

- **msg::register httpd**
This is the HTTPD type for the `core.dda` configuration. If this

Messaging Server is receiving messages forwarded from another Messaging Server, defines the URLHANDLER location on which to look for messages

- **msg::register coreq**

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

The parameters are summarized below:

- TYPE of QUEUE defines Messaging Server location and polling values for picking up messages places in the `/data/<queue>` named by DIR.
- DIR defines the full path of the `/data/core` location. This is set at installation time.
- USE defines where the routing information for each TO label is located.
- POLL and COUNT establish the polling interval and post quantity for the Messaging Server, which determines how often and how many messages are posted at a time. To adjust this, see the topic Configuring the Poll Interval and Post Quantity on page 106.
- 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 disposal location of `/dev/null`.



Default processing of Management Portal data has changed with Messaging Server Version 3.1 and CORE.DDA.CFG Version 3.1 to being re-routed into its own queue (rmpq) This is discussed below and on page 110.

As of Messaging Server version 3.1, the corerouter section is configured to re-queue Management Portal messages into their own queue (rmpq). This

permits separate throttling of the CORE messages being posted to the Management Portal directory from the CORE messages being posted to an ODBC database. For more information, see About the Management Portal Data Queue (rmpq) in CORE.DDA.CFG on page 110.

▶ 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 96.

▶ 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 specifies another Messaging Server when using store and forward, or the URL for a Management Portal or Inventory Manager server.

▶ HP recommends using `msg::register COREODBC <USE type of COREODBC>` to post core data directly to the back-end inventory database. This delivery option has substantial performance benefits over posting the same data to the Inventory Manager server using HTTP, which then posts the data to the back-end 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 106 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 Size and Number for more information. See for Configuring the Log Level, Log Size and Number on page 107 details.

ODBC Settings for CORE, INVENTORY and WBEM Objects

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

DSN	Specify the Data Source Name (DSN) for the Inventory ODBC database. Enclose the entry in quotes.
USER	Specify the user name for the Inventory database identified in the DSN parameter. Default value is
PASS	Specify the password for the user of the Inventory ODBC database. When modifying a password entry, obtain an encrypted entry using the procedure To encrypt a password entry for a database DSN in a configuration file on page 104.
DSN_ATTEMPTS	Number of attempts to connect to the Inventory Manager database DSN. Default is 1.
DSN_DELAY	Delay in seconds between attempts to connect to the Inventory Manager database DSN. Default is 5 seconds.
DSN_PING	Delay in seconds between pinging 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 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 Determines when SQL tables are created and SQL scripts are loaded into memory.

Default is 0, which performs these SQL tasks when the first message is posted. HP recommends using this setting whenever possible because it allows only the necessary commands to be loaded and is a more efficient use of resources.

Set **STARTUPLoad** to 1 to have the SQL tasks performed upon Messaging Server and Data Delivery Agent startup; use this setting when it is necessary to create the SQL tables upon startup due to table-creation limitations set by a Database Administrator.

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

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

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

About the Sections in the INVENTORY.DDA.CFG File

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

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

Queue folder for inventory messages: `<RCS folder>\data\inventory`

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

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

Several configurations are possible.

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

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

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

- **msg::register httpd**
This is the HTTPD type for the `inventory.dda` configuration. If this Messaging Server is receiving messages forwarded from another Messaging Server, defines the URLHANDLER location on which to look for messages (separate locations are specified for core and inventory messages).

- **msg::register inventoryq**
This is the QUEUE type for the `inventory.dda` configuration. Defines how the Messaging Server handles the messages placed by QMSG in the `/data/inventory` folders.

The parameters are summarized below:

- TYPE of QUEUE defines Messaging Server location and polling values for picking up messages places in the `\data\<queue>` named by DIR.
- DIR defines the full path of the `/data/inventory` location. This is set at installation time.
- USE defines where the routing information for each TO label is located.
- POLL and COUNT establish the polling interval and post quantity for the Messaging Server, which determines how often and how many messages are posted at a time. To adjust this, see the topic *Configuring the Poll Interval and Post Quantity* on page 106.
- Retry Attempts (DELAY and ATTEMPTS), after maximum retry attempts, a message is automatically discarded

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

`-To INVENTORY.ODBC`

- **msg::register inventoryodbc**
This is the COREODBC type for the `inventory.dda` configuration. Defines an DSN, User, and Password to post inventory data directly to an ODBC database. For details on the entries, see the topic *ODBC Settings for CORE, INVENTORY and WBEM Objects* on page 96.



This section may be configured during the install.

- **msg::register <USE types of HTTP>**
The section labeled `msg::register inventoryforward` is an example of an HTTP section in the `inventory.dda` module.

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

The URL value specifies another Messaging Server when using store and forward, or the URL for an 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 the 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 106 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 Size and Number* for more information. See for *Configuring the Log Level, Log Size and Number* on page 107 details.

About the Sections in the `WBEM.DDA.CFG` File

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

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

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

Queue folder for `wbem` messages: `<RCS folder>\data\wbem`

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

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

Several configurations are possible.

- Wbem data can be forward to another messaging server. This option is used to place the objects as close to the SQL database as possible before ODBC posting to avoid slow network response.
- Wbem object data can be routed using ODBC directly to the back-end SQL database. This option is best used when the database is close to the current location.
- Wbem data can be routed using HTTP to an Inventory Server. From there, the Inventory Server can post the messages to the back-end 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 `wbem.dda.cfg` file has the following main sections after the header and required line:

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

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

The parameters are summarized below:

- TYPE of QUEUE defines Messaging Server location and polling values for picking up messages.
- DIR defines the full path of the `/data/wbem` location. This is set at installation time.
- USE defines where the routing information for each TO label is located.
- POLL and COUNT establish the polling interval and post quantity for the Messaging Server, which determines how often and how many messages are posted at a time. To adjust this, see the topic [Configuring the Poll Interval and Post Quantity](#) on page 106.
- 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 an ODBC database.

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

Also defines switches, such as `STARTUPLOAD` and `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 96.

- **msg::register <USE types of HTTP>**
The section labeled `msg::register wbemforward` is an example of an HTTP section in `wbem.dda.cfg`.

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

The URL value specifies another Messaging Server when using store and forward, or an 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 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 106 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 107 for more information.

About the Sections in the PATCH.DDA.CFG File

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

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

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

- **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 107. The log files are located in the Logs directory of the root `MessagingServer` installation directory.

- **msg::register patchq**

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

The parameters are summarized below:

- `TYPE` of `QUEUE` defines a Messaging Server location and polling values for picking up messages.
- `DIR` defines the full path of the `/data/patch` message location. This is set at installation time.
- `USE` defines where the routing information for the `TO PATCH` objects are located.
- `POLL` and `COUNT` establish the polling interval and post quantity for the Messaging Server, which determines how often and how many messages are posted at a time. To adjust this, see the topic *Configuring the Poll Interval and Post Quantity* on page 106.
- `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 (unqualified patch objects)

-To PATCH.PREFACE

-To PATCH.ZOBJSTAT

- **msg::register patchfilter**

Appends a filter value to unqualified patch objects before they are rerouted through the msg::register patchrouter section before patchodbc processing.

FILTER defines a value for filtering unqualified patch objects. The default FILTER value is ZOBJCLAS.

- **msg::register patchodbc**

Defines an DSN, User, and Password to post patch data directly to an ODBC database. For details on the entries, see ODBC Settings for PATCH Objects below.



This section may be pre-configured during the install.

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

Note that these options apply for each Worker assigned to process the specific queue. See Configuring the Log Level, Log Size and Number on page 107 for more information.

ODBC Settings for PATCH Objects

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

DSN	Specify the Data Source Name (DSN) for the Patch ODBC database. Enclose the entry in quotes.
USER	Specify the user name for the Inventory database identified in the DSN parameter. Enclose the entry in quotes. Default value is {sa}.
PASS	Specify the password for the user of the Patch ODBC database. Enclose the entry in quotes.

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

The PASS value in the in all the `.cfg` files where specification of DSN parameters is necessary has to be encrypted. When the value is entered during the install process, the installation program takes care of encryption. If you need to modify the password, you can use the `nvdkit` utility to create

an encrypted password, and specify this encrypted value within the appropriate section of the configuration file. Enclose the encrypted value in quotation marks.

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

Additional Tuning Topics

Configuring the Poll Interval and Post Quantity

By default, the Messaging Server is configured to poll the queue location every 10 seconds, and post up to 100 objects at a time. To change the poll interval, modify the `POLL` parameter in the appropriate configuration file and `msg::register` section with a `TYPE` of `QUEUE`.

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

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

Configuring for Retry Attempts

The Messaging Server and the Data Delivery Agents are configured to retry any message that fails to post. By default, the Messaging Server will retry posting the message every hour, and make up to 200 attempts. These values are defined by the `DELAY` and `ATTEMPTS` entries in the sections of the configuration files that have a `TYPE` of `QUEUE`. See Table 9 on page 112 for a list of `msg::register` sections used to modify `QUEUE` processing.



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

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

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

Configuring for Failover

You can configure the Messaging Server to have a one or more servers defined for failover support when defining `HTTP` types. When defining failover servers, each one is assigned a `PRI` value. If the Messaging Server

fails to connect with the first server (that is, the server with the lowest PRI value) it will try the next server on the list (or, the next higher PRI value).

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

To set failover in a *.dda.cfg file

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

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

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

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

Configuring the Log Level, Log Size and Number

The log files for the Messaging Server (`rms.log`) resides in the Logs folder of the `MessagingServer` directory. For example:

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

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

or

```
\data\default
```

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

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

```
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 Messaging Server to Discard or Drain Messages

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

Example: Discarding messages for the Management Portal

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

```

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

```

Example: Draining a Message Queue

As another example, to quickly drain an entire queue, temporarily replace `USE router` in the `msg::register` section for the queue with `USE /dev/null`. See Table 9 on page 112 for a list of the configuration files and

sections that control each queue type. After draining the queue, reset it back to `USE router`.

Configuring the Messaging Server to Route RMP Messages

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

As of Messaging Server V 3.1, the default `core.dda.cfg` configuration will re-queue CORE messages that are to be posted to the Management Portal directory into its own queue, named `rmpq`. This was done to allow for separate throttling of the CORE messages being posted via HTTP to the Management 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/mp
    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:3466/proc/xml/obj
    }
}
```

Restoring Routing for Management Portal Messages

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

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

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

```
ROUTE      {
    TO      CORE.RMP
    USE     /dev/null
```

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

```
TO      CORE.RMP
USE     rmpq
```

- 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 Management Portal server. For example, a Management Portal with a hostname of `RMPSVR` on port `3466` is defined with the following URL entry:

```
URL      http://RMPSVR:3466/proc/xml/obj
```

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

- 6 Save your changes and restart the Messaging Server. For details, see [Starting and Stopping the Messaging Server](#) on page 65.

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 client connect periods if resources are in contention, or if you know a target server is down.

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

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

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

Table 9 Configuration File and Section Used to Modify Queue Processing

Queue	Configuration File in MessagingServer\etc folder	msg::register section
\data\core	core.dda.cfg	msg::register coreq
\data\inventory	core.dda.cfg	msg::register inventoryq
\data\patch	patch.dda.cfg	msg::register patchq
\data\wbem	wbem.dda.cfg	msg::register wbemq
\data\default (older RMS)	rms.cfg	msg::register default

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

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

```
msg::register wbemq {
```

```

        TYPE      QUEUE
        DIR      C:/Novadigm/ConfigurationServer/data/wbemq
        USE      router

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

```

- 4 Save your changes and exit the editor.

To enable a disabled queue

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

Modifying the Priority in which Messages are Processed



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

To modify the priority in which messages are processed, see the earlier topic [How Priority Establishes Messaging Server Processing Order](#) on page 82.

4 Troubleshooting

At the end of this chapter, you will:

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

Troubleshooting the Messaging Server

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

Problem: Log indicates no route defined or failed delivery

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

```
Warning: router1: To: default, From: <rcs>@<rcs_hostname>, subject: - no route defined for
default
Error: MSG/QUEUE: q2: To: default, From: <rcs>@<rcs_hostname>, subject: - failed to deliver to
default
```

Solution:

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

The solution is to review the QMSG calls in ZTASKEND and add any missing -to parameters or missing values. As delivered, QMSG -to values include: -to core.rim, -to core.rmp, -to inventory, and -to inventory.wbem, although there may be others. For details, see the QMSG Method Syntax on page 80.

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 Radia Integration Server (RIS). 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 RIS. 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 RIS 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 RIS service.

If you have any questions regarding this document or this code, please refer to the HP OpenView support web site.

Summary

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

A Optional Messaging Server Configurations

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

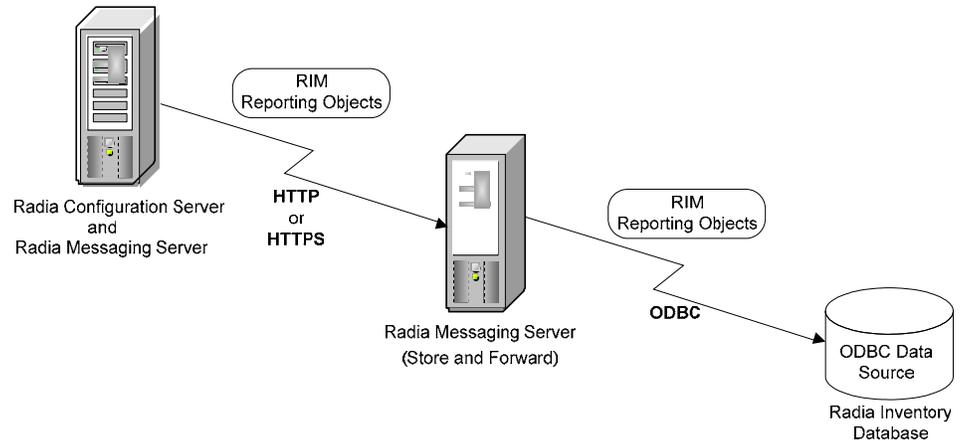
- Example 1: Configuring the Messaging Server for Store and Forward on page 120.
- Example 2: Configuring the Messaging Server to Route Objects from a Single `\data\default Queue`.
- Example 3: Configuring Messaging Server to Route to Multiple Queues on page 133.

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

Example 1: Configuring the Messaging Server for Store and Forward

The Messaging Server includes store and forward capabilities that allow you to create a multiple-Messaging Server environment. When a Messaging Server is used for store-and-forward processing, messages are forwarded from one Messaging Server to another, before being sent to their final destination. This is illustrated in Figure 5 below.

Figure 5 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 RMS to RMS. The forwarding messaging servers route data using HTTP or HTTPS.

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

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

- Installing and Configuring a "Receiving" Messaging Server begins on page 121.

- Configuring a Messaging Server to Forward Messages to another Messaging Server begins on page 125.

Installing and Configuring a "Receiving" Messaging Server

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

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

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



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

About the RMS Receiver

The RMS 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

There is only one listening port specified for an RMS instance. The Port specification is made in `rms.cfg`. Each of the `.dda` modules can specify separate `USE` and `URL` parameters for each of the types of messages they deal with. This is an example of the listener for the `CORE` object messages.

This default configuration is in the `core.dda.cfg`:

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

To specify more than one pair of queue and URL entries, you can code an additional URLHANDLER.

If we assume the RMS server name is TESTSERVER and the listening Port is 3461, then the previous configuration allows messages sent with the following URL to be placed in the queue identified as coreq.

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

By default the coreq is defined as:

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

so the messages received on the url:

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

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

Running the Installation for a Receiving Server and DDAs

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

For example, when prompted for the RMS Message Directories to Scan, you could type:

C:\MessagingServer\data\default

And for the Core Message Directory to Scan, you could type:

C:\MessagingServer\data\core

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 must use forward slashes (/).

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

- Once started, the receiving Messaging Server queues messages sent to it in the newly created "Message Directory to Scan" location. The Messaging Server and DDAs will scan and process these messages as specified by the appropriate data delivery agent configuration file.
- Review the Data Delivery Agent configuration file(s) for the appropriate routing of the messages being received. See the topics in the chapter *Configuring and Tuning the Messaging Server* on page 71.

Configuring a Messaging Server to Forward Messages to another Messaging Server

About Forwarding Messages to a Receiving Messaging Server or DDA

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

Let's assume the RMS was installed with DDAs for each of the Inventory objects (CORE, INVENTORY, and WBEM DDAs), and we want to forward those messages to another RMS installed with those DDAs listening downstream. We want the DDAs on the downstream RMS 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 RMS, we want to tell the DDA to route them using HTTP to another DDA on a downstream server. The needed changes are given in the following procedures.

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

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

- 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.

- 1 Stop the RMS service.
- 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 84.

- 3 Locate the TYPE ROUTER section in the *.cfg file. For example: 'msg::register corerouter' is the TYPE ROUTER section in the core.dda.cfg file. For a data type that is being routed to a section with TYPE *ODBC, switch the USE parameter to the label for a { TYPE HTTP } section.

Table 10 on page 127 shows the USE parameter being switched from coreodbc to coreforward. This reroutes the CORE.ODBC data from a { TYPE COREODBC } section to a {TYPE HTTP } section.



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

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

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

Table 10 on page 127 shows the modifications made to the coreforward and corerouter sections to forward the CORE.ODBC data to a downstream server named DOWNSTREAM.

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

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

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

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

- 5 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 RMS 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://<RMS_hostname:port>/proc/rim/default
```

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

```
URL    http://<RMS_hostname:port>/proc/xml/obj
```

Where:

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

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

- 6 Save the changes to the configuration files that were edited. On the downstream server hosting the RMS, the RMS 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 RMS Receiver on page 122.
- 7 Restart the RMS Service. For details, see Starting and Stopping the Messaging Server on page 65.

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

Messaging Server versions prior to 3.0 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 as of Messaging Server v3.0. It is still supported for customers who are not migrating to the use of multiple queue locations and data directory objects.

- **msg::register default (optional)**
Defines how the Messaging Server handles the messages placed by QMSG in the `/data/default` folder (or, the `/data/default` queue).
- The parameters are defined in About the Sections in the CORE.DDA.CFG File, and summarized below:
 - DIR defines the full path of the `/data/default` location. This is set at installation time.
 - USE defines where the routing information for each TO label is located.
 - POLL and COUNT establish the polling interval and post quantity for the Messaging Server, which determines how often and how many messages are posted at a time. To adjust this, see the topic Configuring the Poll Interval and Post Quantity on page 106.
 - 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
```



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 106.
 - Configure the section to Discard Messages. See *Configuring the Messaging Server to Discard or Drain Messages* on page 109.
- (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 107 for details.

Configuring the Register Default Section

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

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

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

Parameter	Default	Definition
WORKERS	1	<p data-bbox="862 216 1430 279">Optional entry. Number of asynchronous, lightweight processes to create for this queue.</p> <p data-bbox="862 289 1438 478">To drain a queue more quickly, we recommend using WORKERS set to 2. A second worker doubles the processing power of a single Messaging Server configuration, with each worker performing a separate POLL and COUNT.</p> <p data-bbox="862 489 1341 552">Using more than 4 WORKERS is NOT recommended.</p> <p data-bbox="862 562 1450 625">Note: Set to -1 (minus 1) to temporarily disable a queue from being processed.</p>

Example 3: Configuring Messaging Server to Route to Multiple Queues

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

In the RMS configuration shown in Figure 6 on page 134, all messages will be placed in the standard location (the directory `C:/Novadigm/ConfigurationServer/data/default`) when they are created by QMSG. We want to have the Messaging Server route these messages into separate message queues before they are processed. In the code sample that follows, we define the routes for `CORE.RIM` and `CORE.RMP` messages to use `queue1` and `queue2` definitions. The new `queue1` and `queue2` definitions direct the `CORE.RIM` messages to the `C:/rim` directory and the `CORE.RMP` messages to the `C:/rmp` directory.

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

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

The following is a sample RMS configuration sorting messages into multiple queues.

Figure 6 Sample RMS 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:/mp  
}
```

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

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

In the example that follows, following modifications are made:

- 1 The `corerouter` section is modified to route each CORE.ODBC message to two DSN's 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 97](#).
- 3 Processing of the second queue basically duplicates the first:
 - a A QUEUE for `coreodbcq2` is defined and routes its messages to `coreodbcq2router`.
 - b A ROUTER named `coreodbc2router` is defined. It routes messages to a COREODBC section named `coreodbc2`.
 - c A COREODBC section is defined with the name `coreodbc2`. Here is where the second DSN is defined and the messages are posted using ODBC.

See Figure 7 which follows for a sample `core.dda.cfg` file configured with these sections.

Figure 7 Sample CORE.DDA.CFG configured to post data to multiple DSNs

#select core.dda.cfg sections modified to route each message to 2 DSN's 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 coreodbcqlrouter {
    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
# refer to RMS Guide for instructions on encrypting password

```

```

msg::register coreodbc2 {
  TYPE      COREODBC

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

```

This is the end of the topics for alternative configurations.

Index

A

- Adapter for SSL, enabling for Messaging Server, 62
- Application Management Profiles, 32
- ATTEMPTS value, 85, 106
- AUTOCREATE
 - in WBEMODBC section, 96

B

- BOOTSTRAP phase, 76
- BuildAlways, 80

C

- calls to QMSG, 76
- CATALOG RESOLUTION phase, 77
- CATALOG_RESO, 77
- client object processing, 72
- CLIENT SELFMAINTENANCE phase, 77
- CLIENT_REPORTIN, 77
- CLIENT_REPORTING phase, 77
- CLISTAT, 20
- CLISTATS object, 79
- configuration files, location, 84
- Core Data Delivery Agent
 - configuration window, 33
 - configuring, 93
 - configuring during install, 46
 - CORE data routing options, 93
 - ODBC Settings, 96
 - routing RMP data, 111
 - routing RMP messages, 111
- CORE message data, 19
- CORE object data, 21
- CORE.DDA.CFG. *See* Core Data Delivery Agent
- CORE.ODBC message, 136
- coreforward, 22

- coreforward section, 126
- COREODBC section type, 88
- corerouter section, 126, 136
- critical objects, 76

D

- Data Delivery Agents, 19
 - configuring, 84
 - during install, 27
 - on a receiving server, 122
 - to forward messages, 125
- defined, 19
- installing, 28
- installing with Messaging Server, 31
- data queue, 110
- Default Message Directory, 33
- DELAY time, 85, 106
- disabled queue, enabling, 113
- disabling message processing, 112
- discarding messages, 109
- DNS host name, 40
- draining messages, 109
- draining the message queue, 110

E

- Error 404 or 500, 116
- ERROR message, failed to deliver to default, 116

F

- failover, 39, 106
 - configuring with HTTP routing, 106
- FILEPOST object, 21, 79
- filepost.tcl, 21
- FILTER section type, 88
- forwarding messages, 125

H

- HTTP section type, 86
- HTTPD section type, 87
- HTTPS
 - configuring for, 62
 - HTTPS_PORT in URL, 63
- HTTPS section type, 87

I

- installation
 - message directories to scan, 33
 - platform coverage, 26
 - select Data Delivery Agents, 31
 - Store & Forward Port, 41
 - task overview, 28
 - UNIX platforms, 26
 - verifying, 67
 - Windows platforms, 26
 - with Store and Forward, 124
- Inventory Data Delivery Agent
 - configuring, 98
 - configuring during install, 50
 - ODBC Settings, 96
 - routing options, 98
- Inventory Manager
 - critical core objects, 76
- Inventory Manager Server, 20
- INVENTORY message data, 20
- INVENTORY.DDA.CFG. *See* Inventory Data Delivery Agent
- inventoryforward, 22
- InventoryQueue, 80

J

- JOBPARM object, 80
- JOBSTAT object, 80
- JOBTASK object, 80

L

- log files
 - log level, changing, 108

- size and number, changing, 108

M

- Management Portal
 - critical core objects, 76
 - discarding messages for, 109
 - routing messages to, 111
- Messaging Server
 - configuring, 84
 - Data Delivery Agents, introduction, 19
 - installing, 26
 - introduction, 16
 - Inventory data, routing options, 20
 - overview, 18
 - processing on the Configuration Server, 17
 - store and forward configurations, 120
 - store and forward, introduction, 22
 - system requirements, 26
 - troubleshooting, 115
 - tuning topics, 106
- meta data files, 19
- msg::register, 131
- multiple DSNs, sample configuration, 136

N

- nvdkit, encrypt password, 97, 105

O

- ODBC section type, 87
- ODBC Settings
 - configuring for ODBC routing, 96
 - for Patch objects, 104

P

- password encryption, 97, 104
- Patch Data Delivery Agent
 - configuring, 103
 - configuring during install, 58
- Patch Manager, post-installation procedures, 61
- PATCH message data, 20
- PATCH.DDA.CFG. *See* Patch Data Delivery Agent
- PATCH_STATUS, 19

- modifying, 61
- PATCHFILTER section type, 89
- PATCHODBC section type, 89
- PID, obtaining from rms.log, 66
- platform coverage, 26
- Poll interval and post quantity, 106
- post-installation procedures, 61
- PRI value, 106

Q

- QMSG, 18, 116
 - called
 - from PATCH_STATUS, 75
 - from ZTASKEND, 75
 - destinations and Data Delivery Agent locations, 81
 - how it formats messages, 72
 - message syntax, 80
 - priority parameter, 82
- QMSG call syntax, 76
- queue
 - disabling, 112
 - draining, 110
 - names on Configuration Server, 19
- QUEUE section type, 85

R

- Radia Messaging Service
 - as a Windows service, 19
 - starting and stopping, 65
- retry attempts, configuring, 106
- RMP IP address, 40
- RMP Por, 41
- rmppq, 110
- rmppqhttp, 110
- rmprouter, 110
- RMS.CFG, 28
 - configuring, 90
 - dda module load statements, 91
 - log configuration, 91
 - log initialization, 91
 - modifying, 89
 - required packages, 91

- routing sections, 92
 - SSL certificate section, 91
- rms.log, 107
 - PID for Radia Messaging Service, 66
- ROUTER section type, 86

S

- section types, 84
- SERVICE RESOLUTION phase, 77
- SQL database, 20
- SSL
 - enabling for Messaging Server, 62
 - HTTPS_PORT, 63
 - SSL_CERTFILE, 63
 - SSL_KEYFILE, 63
- STARTUPLoad, 21, 97
 - in COREODBC section, 88
- Store and Forward, 22, 120
 - about the receiving server, 121
 - about the sending server, 125
 - configuring a forwarding server and DDAs, 125
 - installing a receiving server and DDAs, 124
 - port number, 41
 - sample CORE.DDA.CFG to forward messages, 127

T

- taskend.tcl, 21
- troubleshooting, 115
- tuning, 106

U

- URLHANDLER, 94, 98, 101, 123
- USE parameter, 109, 126

V

- verify installation, 67

W

- WARNING, no route defined for default, 116
- Wbem Data Delivery Agent, 53
 - configuring, 100

- configuring during install, 53
- ODBC Settings, 96
- routing options, 100
- WBEM message data, 20
- WBEM.DDA.CFG. *See* Wbem Data Delivery Agent
- WBEMAUDT object, 79
- wbemforward, 22
- WBEMODBC section type, 89
- Windows service, 19
- WORKERS, how to disable, 112

X

- XML files, 19

Z

- ZERROR, 77
- ZERRORM, 77
- ZMTHPRMS
 - queue value, 37
- ZMTHPRMS, modifying queue name for patch, 61
- ZOBJSTAT, 20
- ZTASKEND, 18, 116
 - about, 75
 - critical core object processing, 76
 - modifying always objects, 79
 - modifying for critical objects, 79
 - RMS 2.x and 3.x processing, 80