HP Client Automation Enterprise

Messaging Server

for the HP-UX, Linux, Solaris, and Windows operating system

Software Version: 7.20

Migration Guide

Document Release Date: July 2008

Software Release Date: August 2008

Documentation Updates

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

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

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

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

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

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

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

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

Table 1 indicates changes made to this document.

Table 1 Document changes

Chapter	Version	Changes
All	7.20	The HP Configuration Management products are newly named HP Client Automation for 7.20. Updated product names throughout the guide.
Documentation Updates	7.20	Updated URL for manuals.
Chapter 2	7.20	Page 9, renamed chapter Upgrading to Messaging Server 7.20, and reorganized topics.
Chapter 2	7.20	Page 9, Changes as of Version 7.20, new topic.
Chapter 2	7.20	Page 11, Upgrading your RCS ZTASKEND REXX Method, Added note: Prior to updating the database, update your ZTASKEND to version 1.18 or above.
Chapter 2	7.20	Page 11, Updates for Inventory Manager Database Version 7.20, summarizes the four tables that require additional columns to support this release: DeviceConfig, HDeviceConfig, SMBiosInfo, and rWin32_DiskDrive.
Chapter 2	5.10	Page 16, Corrected Oracle syntax in the procedures to update Inventory Database from Version 5.10.
Appendix A	7.20 Aug 2008	Page 29, Inventory Database Schema for Version 7.20 adds this script and table: win32_physicalmemory.sql creates the rWin32_PhysicalMemory table; also adds a set of scripts and tables for Vulnerability Management.

Support

You can visit the HP Software support web site at:

www.hp.com/go/hpsoftwaresupport

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

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

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

Most of the support areas require that you register as an HP Passport user and sign in. Many also require a support contract.

To find more information about access levels, go to:

www.hp.com/managementsoftware/access_level

To register for an HP Passport ID, go to:

www.managementsoftware.hp.com/passport-registration.html

Contents

1 Introduction

About this Guide

Who this Guide is for

This migration guide is for system administrators who want to upgrade the Messaging Servers in their HP Client Automation (HPCA) environment to Version 7.20.

This guide contains information for the Windows and UNIX platforms, and for migrating from either:

- Messaging Server v5.1x or v5.00
- Messaging Server v2.x or v3.x

You should be familiar with HPCA infrastructure products such as the Configuration Server and Database, methods such as ZTASKEND, the SQL Database used by the Inventory Manager application and the Reporting Server. If using Patch Manager, you should be familiar with that product and SQL Database.

For details, see the appropriate guides for each product.

2 Upgrading to Messaging Server 7.20

Overview

Use the following procedures to upgrade from an existing Radia Messaging Server 2.x, 3.x or Configuration Management Messaging Server 5.xx environment to an HP Client Automation Messaging Server 7.20 environment.

The Messaging Server 7.20 release is a drop-in replacement for previously released versions of the Messaging Server that have Data Delivery Agent (.dda) support. The upgrade allows you to use existing configuration files and any customized scripts that you have for mapping client object data into backend databases.

Changes as of Version 7.20:

If you are upgrading from 5.1x to 7.20, you will see the following changes:

- **HP Client Automation Rebranding:** As of Version 7.20, the Messaging Server has been renamed from the HP Configuration Management Messaging Server to the HP Client Automation (HPCA) Messaging Server. The default installation paths remain the same:

  `C:\Program Files\Hewlett-Packard\CM\MessagingServer` for Windows

  `/opt/HP/CM/MessagingServer` for UNIX

- **SQL Schema Changes**: Several tables in the Inventory SQL Database (Core and Inventory tables) require additional columns in order to support Client Automation 7.20 features, such as Vulnerability Management and Thin Client support. There is also a new table.

  Prior to upgrading the Messaging Server to 7.20, all customers must apply the table changes to the Inventory Manager Database. Refer to Updates for Inventory Manager Database Version 7.20 on page 11.

- **Vulnerability Support embedded in Core.DDA**: The Messaging Server and the core.dda from this Version 7.20 release are required for Vulnerability Management, as well as the latest ZTASKEND REXX version on the HPCAE 7.20 media; Vulnerability Management uses the same ODBC DSN connections as the Core.DDA to post a device's OVAL-based security and vulnerability status to the appropriate Inventory Database tables. Refer to the *HP Client Automation Enterprise Manager User Guide* for more information on obtaining Vulnerability Reports for your managed-devices.

Changes as of Version 5.10:

If you are upgrading from 5.00 to this release, you will see the following changes introduced as of Version 5.10:

- The Messaging Server install includes an optional selection for a Usage Manager DDA. This Data Delivery Agents offers support for the collation of Application Usage Manager files. Refer to the *Application Usage Manager User Guide* for more information.

- Two additional scripts are provided to migrate an Inventory Manager Database to Unicode. If you already modified your database to Unicode for version 5.00, you need to apply the last two steps. Refer to page 27 for SQL, and refer to page 28 for Oracle. These required database changes apply to all customers upgrading from Version 5.00.
- Patch Manager 5.10 included an additional reporting object: PASTATUS. If your CM Messaging Server uses a 'Patch Message Directory to Scan' queue that is not named `patch`, refer to Verify the Patch Method Connections and Queue Name on page 18.

Changes as of Version 5.00:

The installation program, as of Messaging Server 5.00, was updated in the following ways:

- The default installation path changed to:

  `C:\Program Files\Hewlett-Packard\CM\MessagingServer` for Windows

  `/opt/HP/CM/MessagingServer` for UNIX

  If desired, the default path can be changed during the installation process to point at an existing Messaging Server path.
- This version adds several scripts needed to generate standard tables for wbem/cim.
- Use of the newest generation of nvdkit allows support for new database character sets that support multiple-languages, such as the nvarchar datatype for SQL Server and nvarchar2 datatype for Oracle. To take advantage of these new datatypes, conversion of the backend database must be performed. Scripts included with this release can be used for this database conversion.
- Datadirect Connect ODBC drivers for the supported Unix and Linus platforms will be installed by default. These ODBC drivers will allow data posting to a database directly from these platforms.

The Messaging Server install program will:

- Create a new `rms.cfg` file (as long as you rename your existing one).
- Create `*.dda.cfg` configuration files for each Data Delivery Agent that is selected during the install. There are five available Data Delivery Agents: CORE, INVENTORY, WBEM, PATCH and USAGE The Data Delivery Agents can be used to post data to a SQL Database or Oracle Database.
- Add the same scripts and `*.sql` code to the Messaging Server that is provided with the (now retired) Inventory Manager Server for creating the SQL tables and to modify the data in the Inventory database.
- The procedures include a post-install task of relocating custom SQL code from an existing Inventory Manager Server to your Messaging Server.

  The procedures include a post-install task of relocating any custom SQL code from your previous Inventory Manager Server to your Messaging Server.

The Messaging Server install program will not:

- Install a Messaging Server fully configured for store and forward capabilities. Following the installation, you need to edit the appropriate configuration files and switch the routing options to forward messages to another Messaging Server. For details, refer to the Store and Forward Configuration topics in Appendix A of the *HPCA Messaging Server Installation and Configuration Guide.*

Upgrading from previous versions (Windows and UNIX)

Upgrading your RCS ZTASKEND REXX Method

Prior to upgrading the Messaging Server, it is a best practice to adopt the latest version of the ZTASKEND REXX delivered on the Client Automation media with the Configuration Server.

It is also a best practice to upgrade the ZTASKEND REXX method before upgrading the Messaging Server.

New ZTASKEND Method in Version 7.20

New features in Client Automation 7.20, such as Vulnerability Management and Windows CE Thin Client management, require that you use the latest version of ZTASKEND that is supplied on the HPCAE 7.20 media. As of this writing, the latest ZTASKEND version is 1.18.

Updates for Inventory Manager Database Version 7.20

Make the following changes to an existing Inventory Manager database to support this release of Messaging Server and Reporting Server. Make these changes whether you are using an Oracle or SQL Server database. These changes are required.

1. DeviceConfig and HDeviceConfig Table Changes: Add columns lastuser, subnetaddr and subnetmask

The DeviceConfig and HDeviceConfig tables require new columns: **lastuser, subnetaddr** and **subnetmask**. Either delete the existing DeviceConfig and HDeviceConfig tables so they are created with the necessary columns by the Messaging Server 7.20 installation, or, modify the existing Table definitions for DeviceConfig and HDeviceConfig by adding the columns listed in the following table:

Column Name	Data Type	Length	Allow Nulls
lastuser	nvarchar (if Unicode) or varchar (if not Unicode)	128	√
subnetaddr	nvarchar (if Unicode) or varchar (if not Unicode)	128	√
subnetmask	nvarchar (if Unicode) or varchar (if not Unicode)	128	√

2. SMBiosInfo Table Changes: Add columns for enc_model, enc_name, enc_slot, rackname

The SMBiosInfo table requires the new columns below. Either delete the existing SMBiosInfo table so it is created with the necessary columns by the Messaging Server 7.20 installation, or, modify the existing Table definition by adding the columns below:

Column Name	Data Type	Length	Allow Nulls
enc_model	nvarchar (if Unicode) or varchar (if not Unicode)	128	√
enc_name	nvarchar (if Unicode) or varchar (if not Unicode)	128	√
enc_slot	nvarchar (if Unicode) or varchar (if not Unicode)	128	√
rackname	nvarchar (if Unicode) or varchar (if not Unicode)	128	√

Use the task approach of your choice, below, to add the new columns.

3. rWin32_DiskDrive Table Changes: Add columns for Vista devices: wSerialNumber, wFirmwareRevision, wSignature and wCapabilityDescriptions2

The rWin32_DiskDrive table requires 4 new columns for Vista devices.

Either delete the existing rWin32_DiskDrive table so it is created with the necessary column by the Messaging Server installation, or, modify the existing Table definition for rWin32_DiskDrive by adding the following columns:

Column Name	Data Type	Length	Allow Nulls
wSerialNumber	nvarchar (if Unicode) or varchar (if not Unicode)	128	√
wFirmwareRevision	nvarchar (if Unicode) or varchar (if not Unicode)	128	√
wSignature	int	4	√
wCapabilityDescriptions2	nvarchar (if Unicode) or varchar (if not Unicode)	128	√

This completes the table changes required for Version 7.20.

Procedures

To modify the table definitions:

SQL Server: Run the following commands against a database running on SQL Server. If your database is not Unicode, substitute varchar for nvarchar:

```
ALTER TABLE DeviceConfig ADD subnetaddr nvarchar(128) NULL; GO
ALTER TABLE DeviceConfig ADD subnetmask nvarchar(128) NULL; GO
ALTER TABLE DeviceConfig ADD lastuser nvarchar(128) NULL; GO

ALTER TABLE HDeviceConfig ADD subnetaddr nvarchar(128) NULL; GO
ALTER TABLE HDeviceConfig ADD subnetmask nvarchar(128) NULL; GO
ALTER TABLE HDeviceConfig ADD lastuser nvarchar(128) NULL; GO

ALTER TABLE SMBiosInfo ADD enc_model nvarchar(128) NULL; GO
ALTER TABLE SMBiosInfo ADD enc_name nvarchar(128) NULL; GO
ALTER TABLE SMBiosInfo ADD enc_slot nvarchar(128) NULL; GO
ALTER TABLE SMBiosInfo ADD rackname nvarchar(128) NULL; GO

ALTER TABLE rWin32_DiskDrive ADD wSerialNumber nvarchar(128) NULL; GO
ALTER TABLE rWin32_DiskDrive ADD wFirmwareRevision nvarchar(128) NULL; GO
ALTER TABLE rWin32_DiskDrive ADD wSignature int NULL; GO
ALTER TABLE rWin32_DiskDrive ADD wCapabilityDescriptions2 nvarchar(128)
```

ORACLE: Run the following commands against a database running on Oracle to add the new columns. If your database is not Unicode, substitute varchar2 for nvarchar2:

```
ALTER TABLE DeviceConfig
 ADD (
   subnetaddr nvarchar2(128) NULL,
   subnetmask nvarchar2(128) NULL,
   lastuser nvarchar2(128) NULL);

ALTER TABLE HDeviceConfig
 ADD (
  subnetaddr nvarchar2(128) NULL,
  subnetmask nvarchar2(128) NULL,
  lastuser nvarchar2(128) NULL);

ALTER TABLE SMBiosInfo
 ADD (
  enc_model nvarchar2(128) NULL,
  enc_name nvarchar2(128) NULL,
  enc_slot nvarchar2(128) NULL,
  rackname nvarchar2(128) NULL);

ALTER TABLE rWin32_DiskDrive
 ADD (
  wSerialNumber nvarchar2(128) NULL,
  wFirmwareRevision nvarchar2(128) NULL,
  wSignature int NULL,
  wCapabilityDescriptions2 nvarchar2(128) NULL);
```

To delete and rebuild the existing tables (and reapply any customizations, if necessary):

1. Stop the Messaging Server service (RMS.TKD).

   Stopping the RMS.TKD service automatically stops message processing for each Data Delivery Agent queue.

2. Create a backup of the `device.config.sql`, `smbios.info.sql`, `win32_diskdrive.sql` and `taskend.tcl` files that are found in the following folders where your Messaging Server is installed.

   `\etc\core\sql\hp\device.config.sql` *(hp-delivered version)*

   `\etc\core\sql\hp\smbios.info.sql` *(hp-delivered version)*

   `\etc\core\sql\hp\win32_diskdrive.sql` *(hp-delivered version)*

   `\etc\core\lib\taskend.tcl`

3. Delete the existing `device.config.sql, smbios.info.sql, win32_diskdrive.sql and taskend.tcl` files from the above locations, so that the newest versions can be unpacked when the Messaging Server is installed and executed.

4. Following installation, reapply any customizations to the default versions of `device.config.sql, smbios.info.sql, win32_diskdrive.sql` and `taskend.tcl` files, which now contain the new columns. Place the customized versions of any `*.sql` file in the `\etc\core\sql` directory. This allows your custom scripts to take precedence over the default scripts in the lower-level `\hp` subdirectory. Also reapply any customizations from your backup version of `taskend.tcl` to the default version of `taskend.tcl`, if appropriate, and place the customized file back in `\etc\core\lib\`.

Updates for Inventory Manager Database as of Version 5.10

The changes below were introduced as of Messaging Server 5.10. If you are upgrading from a pre-5.10 version of the Inventory Manager database, also make these changes to support this release of Messaging Server and Reporting Server. Make these changes whether you are using an Oracle or SQL Server database. These changes are required.

1. DeviceConfig and HDeviceConfig Table Changes: Add columns clientrel and tpmchip

The DeviceConfig and HDeviceConfig tables require new columns: **clientrel** and **tpmchip**. Either delete the existing DeviceConfig and HDeviceConfig tables so they are created with the necessary column by the Messaging Server installation, or, modify the existing Table definitions for DeviceConfig and HDeviceConfig by adding the following columns:

Column Name	**Data Type**	**Length**	**Allow Nulls**
clientrel	nvarchar (if Unicode) or varchar (if not Unicode)	128	√
tpmchip	nvarchar (if Unicode) or varchar (if not Unicode)	1	√

2. SMBiosInfo Table Changes: Add columns for biosvend, biosdate, biosvers, flashmem

The SMBiosInfo table requires the new columns below. Either delete the existing SMBiosInfo table so it is created with the necessary columns by the Messaging Server installation, or, modify the existing Table definition by adding the columns below:

Column Name	**Data Type**	**Length**	**Allow Nulls**
biosvend	nvarchar (if Unicode) or varchar (if not Unicode)	128	√
biosdate	nvarchar (if Unicode) or varchar (if not Unicode)	128	√
biosvers	nvarchar (if Unicode) or varchar (if not Unicode)	128	√
flashmem	nvarchar (if Unicode) or varchar (if not Unicode)	128	√

Use the task approach of your choice, below, to add the new columns.

Procedures

To modify the table definitions:

SQL Server: Run the following commands against a database running on SQL Server. If your database is not Unicode, substitute varchar for nvarchar:

```
ALTER TABLE DeviceConfig ADD clientrel nvarchar(128) NULL; GO

ALTER TABLE HDeviceConfig ADD clientrel nvarchar(128) NULL; GO

ALTER TABLE DeviceConfig ADD tpmchip nvarchar(1) NULL; GO

ALTER TABLE HDeviceConfig ADD tpmchip nvarchar(1) NULL; GO

ALTER TABLE SMBiosInfo ADD biosvend nvarchar(128) NULL; GO

ALTER TABLE SMBiosInfo ADD biosdate nvarchar(128) NULL; GO

ALTER TABLE SMBiosInfo ADD biosvers nvarchar(128) NULL; GO

ALTER TABLE SMBiosInfo ADD flashmem nvarchar(128) NULL; GO
```

ORACLE: Run the following commands against a database running on Oracle to add the new columns. If your database is not Unicode, substitute varchar2 for nvarchar2:

```
ALTER TABLE deviceconfig
 ADD (
  clientrel nvarchar2(128) NULL,
  tpmchip nvarchar2(1) NULL);

ALTER TABLE hdeviceconfig
 ADD (
  clientrel nvarchar2(128) NULL,
  tpmchip nvarchar2(1) NULL);

ALTER TABLE SMBiosInfo
 ADD (
  bisovend nvarchar2(128) NULL,
  biosdate nvarchar2(128) NULL,
  biosvers nvarchar2(128) NULL,
  flashmem nvarchar2(128) NULL)
```

To delete and rebuild the existing tables (and reapply any customizations, if necessary):

Follow the procedures to delete and rebuild existing tables given on page 14.

This completes the topics for upgrading the Inventory Database tables for Messaging Server 7.20.

Upgrading from Messaging Server 5.xx

Inventory Database Migration and Updates

- If you previously converted your Inventory Database to Unicode as part of migrating your Messaging Server to Version 5.00, run the final database migration script prior to upgrading the Messaging Server to Version 5.10. Refer to page 27 for SQL Server, or page 28 for Oracle.
- All customers also need to apply changes to the Inventory Manager Database prior to upgrading the Messaging Server. Refer to page on page 11.

1 Stop the HP OpenView CM Messaging Server service (RMS.TKD).

   Stopping the service automatically stops message processing for each Data Delivery Agent queue.

2 Create a backup of the directory where your existing Messaging Server is installed.

3 The installation program will upgrade the Messaging Server and Data Delivery Agent modules, but does not replace the associated configuration files with the new ones. Optionally, delete an existing configuration file prior to running the install to obtain the newest default configuration for it:

```
rms.cfg
core.dda.cfg
inventory.dda.cfg
wbem.dda.cfg
patch.dda.cfg
```

4 Launch the installation program for the Messaging Server, available from the following platform-specific location on the HPCA 7.20 media:

```
Infrastructure\extended_infrastructure\messaging_server\<platform>
```

— For Windows, click on **setup.exe** to launch the installation program.

— For a UNIX platform, enter the following command:

```
setup
```

and press **Enter**.

Follow the prompts to complete the installation, making sure to select the installation of all existing data delivery agents.

This completes the steps to apply Messaging Server 7.20 to an existing 5.xx installation.

Post-Installation Considerations

Patch Manager: Set the DBTYPE for a Patch ODBC Database on Oracle

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

To change the DBTYPE for ORACLE:

1 Use a text editor to edit the `patch.dda.cfg` file located in the \etc folder of where the Messaging Server was installed.

2 Locate the `patchddaodbc` section, and set the `DBTYPE` to **"ORACLE"**. Enclose the value in quotes. An example is shown below:

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

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

Verify the Patch Method Connections and Queue Name

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

  PATCH_DEERROR

  PATCH_BUSTATUS

  PATCH_DESTATUS

  PATCH_RESTATUS

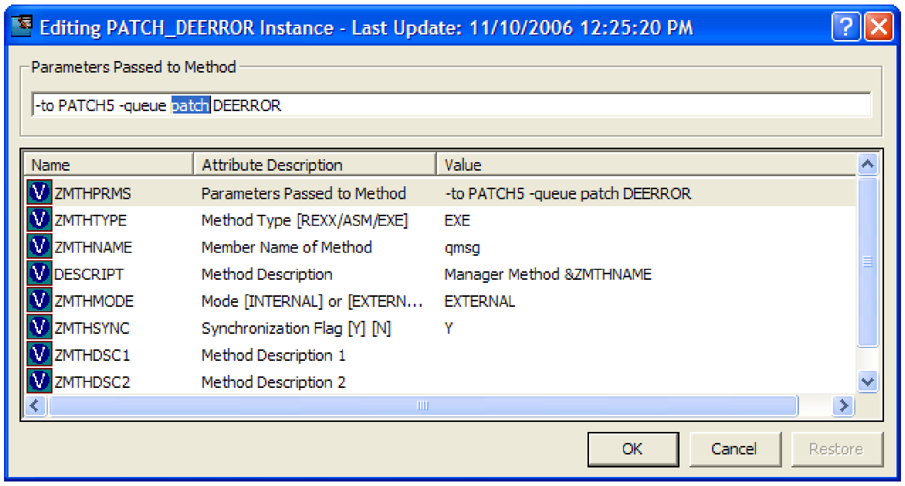
  PATCH_PASTATUS

To modify the queue name in the five PATCH_* methods

1 Use the CM Admin CSDB Editor to edit the ZMTHPRMS attribute of the PRIMARY.SYSTEM. ZMETHOD.PATCH_DEERROR instance, as shown in Figure 1on page 19.

2 Adjust the –queue patch value to reflect the directory named as the "Patch Message Directory to Scan".

Figure 1 Specify the Patch queue name in ZMTHPRMS.


Editing PATCH_DEERROR Instance - Last Update: 11/10/2006 12:25:20 PM
Parameters Passed to Method
OK
Cancel
Restore

-to PATCH5 -queue patch DEERROR

Name	Attribute Description	Value
ZMTHPRMS	Parameters Passed to Method	-to PATCH5 -queue patch DEERROR
ZMTHTYPE	Method Type [REXX/ASM/EXE]	EXE
ZMTHNAME	Member Name of Method	qmsg
DESCRIPT	Method Description	Manager Method &ZMTHNAME
ZMTHMODE	Mode [INTERNAL] or [EXTERN...	EXTERNAL
ZMTHSYNC	Synchronization Flag [Y] [N]	Y
ZMTHDSC1	Method Description 1	
ZMTHDSC2	Method Description 2	

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

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

to

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

3 Save your changes.

4 Repeat the above change you made to the ZMTHPRMS –queue value in the PATCH_DEERROR instance to these PRIMARY.SYSTEM methods:

PATCH_BUSTATUS

PATCH_DESTATUS

PATCH_RESTATUS

PATCH_PASTATUS

5 Save your changes.

Using Store and Forward Configurations (to Place Objects Close to a SQL-compliant Database)

As previously mentioned, the installation program does not install a Messaging Server fully configured for store and forward capabilities. Following the installation, you need to edit the appropriate configuration files and switch the routing options to forward messages to another Messaging Server.

For details, refer to the Store and Forward Configuration topics in the *HPCA Messaging Server Installation and Configuration Guide.*

Using the HP-Supplied Datadirect Connect ODBC drivers

Messaging Servers installed on UNIX platforms require the configuration of the HP-provided Datadirect Connect drivers in order to post data to a backend database. For information on how to configure these drivers, refer to the topics in the *Messaging Server Installation and*

Configuration Guide, located in the `\documentation` folder of the HP Client Automation 7.20 media.

Upgrading from Messaging Server 2x or 3.x

HP recommends performing this upgrade after you have upgraded the ZTASKEND method on the Configuration Server to version 1.12 or above. See Upgrading your RCS ZTASKEND REXX Methodon page 11.

CM Messaging Server 7.20 requires the Data Delivery Agents delivered with Version 7.20, and vice versa, as well as the latest version of nvdkit. All of these modules are installed by default. Be sure to re-install each Data Delivery Agent that was previously installed so that the Data Delivery Agent modules are updated as well as the Messaging Service module.

1 Stop the Messaging Server (RMS) service (RMS.TKD).

   Stopping the RMS service automatically stops message processing for each Data Delivery Agent queue.

2 Create a backup of the directory where your existing Messaging Server is installed.

3 Delete the following subdirectories from the `etc` directory of where the Messaging Server is installed:

```
/etc/core/sql
/etc/core/lib
/etc/inventory/sql
/etc/inventory/lib
/etc/wbem/sql
```

4 The installation program will upgrade the Messaging Server and Data Delivery Agent modules, but does not replace the associated configuration files with the new ones. Optionally, delete or rename an existing configuration file prior to running the install to obtain the newest default configuration for it:

```
rms.cfg
core.dda.cfg
inventory.dda.cfg
wbem.dda.cfg
```

5 CM Patch Manager object processing changed for Version 5.00, and requires the new Patch DDA configuration file. Rename or delete an existing `patch.dda.cfg` file prior to running the install.

6 Launch the installation program for the CM Messaging Server, available from the following platform-specific location on the HPCA 7.20 media:

```
Infrastructure\extended_infrastructure\messaging_server\<platform>
```

— For Windows, click on **setup.exe** to launch the installation program.

— For a UNIX platform, enter the following command:

```
setup
```

and press **Enter**.

Follow the prompts to complete the installation, making sure to select the installation of all existing data delivery agents.

7 Following installation, reapply any customizations to the `*.SQL` and `*.TCL` files located in the `\etc\core`, `\etc\inventory` and `\etc\wbem` directories.

   The Data Delivery Agents for Messaging Server unpack the default versions of the `.sql` files into subdirectories named `\etc\<dda module>\sql\hp`.

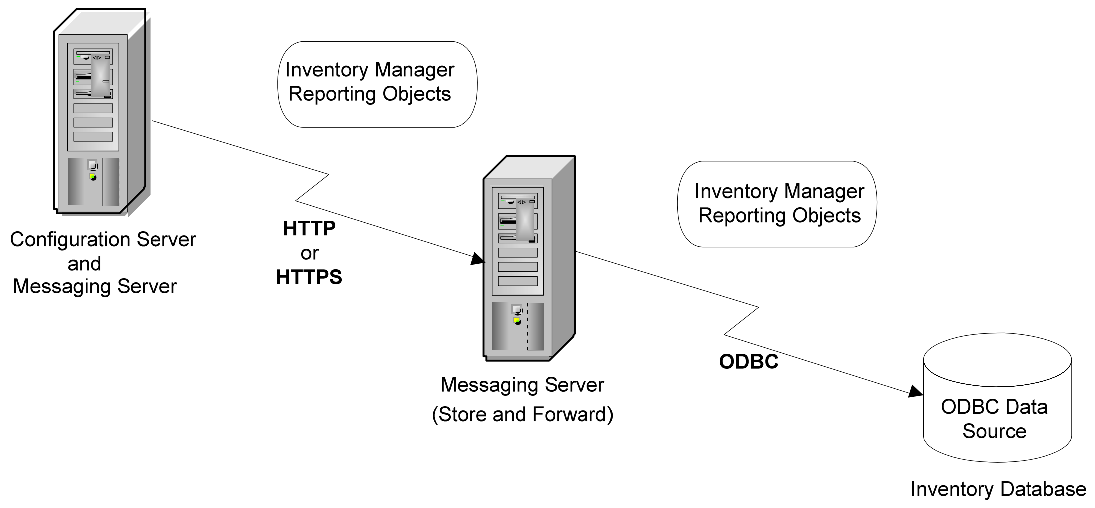
This allows your custom scripts to be placed in the `\etc\<dda module>\sql` directories and take precedence over the default scripts in the lower-level `\hp` subdirectory.

8 If your Messaging Server 3.x environment used an Inventory Manager Server, the next step is to port any customizations you made from the existing Inventory Manager Server to the appropriate Messaging Server locations. See Migrating Custom SQL Code from an Inventory Manager Server below.

This completes the steps to apply CM Messaging Server 5.10 to an existing 3.x installation.

HP-Recommended Best Practices for Messaging Server 5.x

The Data Delivery Agent modules for CORE, INVENTORY, and WBEM objects allow for posting of these objects directly into a back-end SQL compliant database using ODBC. HP recommends that the Messaging Server used to post these objects via ODBC be placed as close to the SQL compliant database as possible to minimize the network response time. Often this means using the Messaging Server co-located with the Configuration Server as a forwarding messaging server, and installing a downstream Messaging Server close to the SQL compliant database that is configured to do the actual ODBC posting. This is illustrated in the following figure.


Inventory Manager Reporting Objects
Configuration Server and Messaging Server
HTTP or HTTPS
Inventory Manager Reporting Objects
Messaging Server (Store and Forward)
ODBC
ODBC Data Source
Inventory Database

Migrating Custom SQL Code from an Inventory Manager Server

If you elected to install any of the Data Delivery Agents for posting CORE, INVENTORY, and WBEM objects to the Inventory Manager using ODBC, the customized versions of any files listed in Table 2 on page 23 can be copied from their locations on your Inventory Manager Server to the equivalent locations on the Messaging Server. Details are given in the following section.

The Data Delivery Agents can be used to post data directly to a SQL Database or Oracle Database.

You only need to port the custom code to a Messaging Server that is being used to post data using ODBC to an Inventory database. It is not necessary to port any customizations to a Messaging Server that is forwarding data to another Messaging Server.

About the Scripts and SQL Queries used with the Data Delivery Agents

The Data Delivery Agents for CORE, WBEM and INVENTORY data post their message data into the same SQL tables created by the previous Inventory Manager Server. These Data Delivery Agents use the exact same table definitions used by the legacy Inventory Manager Server to create tables, update 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 definitions for these tables and associated SQL queries (as delivered from HP) are contained in the `/etc/<module name>/sql/hp` directories. However, custom versions of these `.sql` files are to be placed in the `/etc/<module name>/sql` directories; this means the customized versions will be executed instead of the HP-delivered versions placed in the lower level `hp` subdirectories.

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 previously used on the Inventory Manager Server. The script necessary to map the INVENTORY object data (FILEPOST object) is called `filepost.tcl`. Both these scripts are found in the `/etc/<module name>/lib` directory of the CM Messaging Server. Using the identical scripts found on the Inventory Manager Server allows previous users of this Infrastructure service to migrate any customized scripts directly into the directory for the associated Data Delivery Agent module.

To migrate custom code from an Inventory Manager Server to a Messaging Server

1 If necessary, stop the service for the Messaging Server.

2 Create a backup of the Messaging Server `\etc` directory before porting any customized code.

3 Stop the [httpd] service for the Inventory Manager Server.

4 Use Table 2 below to locate the appropriate Messaging Server directory and file for each object type. Copy any customized versions of the code on your Inventory Manager Server to the appropriate location on the Messaging Server.

Table 2 Directory Locations for Migrating Custom Code from RIM to RMS

Data Directory Agent: and Files	Inventory Server Directory Location	Messaging Server Directory Location
core.dda support:		
`taskend.tcl`	*<RIS>*`\etc\rim\lib`	*<RMS>*`\etc\core\lib`
`*.sql files`	*<RIS>*`\etc\sql`	*<RMS>*`\etc\core\sql`
inventory.dda support		
`filepost.tcl`	*<RIS>*`\etc\rim\lib`	*<RMS>*`\etc\inventory\lib`
`*.sql files`	*<RIS>*`\etc\sql`	*<RMS>*`\etc\inventory\sql`
wbem.dda support		
`*.sql files`	*<RIS>*`\etc\sql\wbem`	*<RMS>*`\etc\wbem\sql`

For example, if you have a customized version of `taskend.tcl` on your Inventory Manager Server, copy it to the `<RMS>\etc\core\lib` location on the Messaging Server.

5. Restart the Messaging Server service or process.
6. If you have used Data Delivery Agents with the ODBC routing options to post the CORE, INVENTORY, and WBEM inventory objects directly to an Inventory database, you do not need to restart the Inventory Manager Server.

Conversion of Database to Unicode Datatypes (Optional)

Modifications must be made to an existing Inventory Database if they were created with varchar datatypes to take advantage of multilingual support available using nvarchar datatypes. This conversion is not required and the Messaging Server 5.00 or above will work with the existing database tables without conversion.

Why convert the database for Unicode support?

Storing data in multiple languages within one database is difficult to manage when you use only character data and code pages. It is also difficult to find one code page for the database that can store all the required language-specific characters. Additionally, it is difficult to guarantee the correct translation of special characters when being read or updated by different clients running various code pages. Databases that support international clients should always use Unicode data types instead of non-Unicode data types.

For example, consider a database of customers in North America that must handle three major languages:

- Spanish names and addresses for Mexico
- French names and addresses for Quebec
- English names and addresses for the rest of Canada and the United States

When you use only character columns and code pages, you must take care to make sure the database is installed with a code page that will handle the characters of all three languages. You must also take care to guarantee the correct translation of characters from one of the languages when read by clients running a code page for another language.

With the growth of the Internet, it is even more important to support many client computers that are running different locales. Selecting a code page for character data types that will support all the characters required by a worldwide audience would be difficult.

The easiest way to manage character data in international databases is to always use the Unicode nchar, nvarchar, and nvarchar(max) data types, instead of their non-Unicode equivalents, char, varchar, and text.

Unicode is a standard for mapping code points to characters. Because it is designed to cover all the characters of all the languages of the world, there is no need for different code pages to handle different sets of characters. SQL Server 2005 supports the Unicode Standard, Version 3.2.

If all the applications that work with international databases also use Unicode variables instead of non-Unicode variables, character translations do not have to be performed anywhere in the system. Clients will see the same characters in the data as all other clients.

SQL Server 2005 stores all textual system catalog data in columns having Unicode data types. The names of database objects, such as tables, views, and stored procedures, are stored in Unicode columns. This enables applications to be developed by using only Unicode, and helps avoid all issues with code page conversions.

About the SQL Server Migration Scripts

There are scripts included in a migrate directory with the Messaging Server install for converting SQL Server and Oracle databases to convert the varchar datatype to nvarchar. All default conversion scripts will address the standard tables created by the Messaging Server and previously by RIM Server. Additional custom tables must be converted separately. See the Appendix on page 29 for the listing of the standard tables created by the included scripts.

All scripts must be reviewed by a Data Base Administrator familiar with the specific custom environment. The scripts are given as guidelines and in some cases must be edited prior to execution.

Always backup your existing database prior to performing this type of conversion.

Possible Unicode Migration Issues and Resolution Options

Expected Warning: Table Index Exceeds the Maximum Number of Bytes

During the migration process, you will see database warnings when running the migration scripts against your Inventory tables. You will be warned whenever the total bytes for a table's composite index, after conversion to the Unicode acceptable datatype, are greater than the permitted maximum size.

Using a SQL Server database as an example, the maximum size allowed for an index is 900 bytes. Several of the Inventory tables, when converted to nvarchar datatypes, will show a warning that the maximum size of the index has been exceeded.

In most cases these warnings do not present a problem because the column sizes were created with default values that were much larger than needed. The warning includes an alert that if subsequent insert or update actions on the variable-type columns result in a total size greater than 900 bytes, the action will fail and the user will get a run-time error. Likewise, if the index definition is composed of variable-type columns only, and the maximum total size of these columns is greater than 900 bytes, SQL Server will create the index, but return a warning.

The tables that can be expected to show this warning are listed below.

rCIM_CDROMDrive	rCIM_SoftwareElement
rCIM_Directory	rCIM_SoftwareFeatureElements
rCIM_DiskDrive	rCIM_StorageVolume
rCIM_DVDDrive	rCIM_UnixLocalFileSystem
rCIM_EthernetAdapter	rCIM_UnixOperatingSystem
rCIM_Export	rNVD_DownloadStatistics
rCIM_HPUX_SwBundles	rNVD_GroupAccount
rCIM_IDEController	rNVD_GroupMember
rCIM_LogicalDisk	rNVD_MulticastStatistics
rCIM_LogicalDiskBasedOnVolume	rNVD_NISGroupAccount

rCIM_MediaPresent	rNVD_NISUserAccount
rCIM_NFS	rNVD_Product
rCIM_OperatingSystem	rNVD_SolarisPatch
rCIM_ParallelController	rNVD_UserAccount
rCIM_Process	rRegistry
rCIM_Processor	rWin32_BIOS
rCIM_ProductSoftwareFeatures	rWin32_ComputerSystemProduct
rCIM_ResidesOnExtent	rWin32_Product
rCIM_SCSIController	rWin32_SoftwareElement
rCIM_SCSIInterface	rWin32_SoftwareFeature
rCIM_ServiceI	rWin32_StartupCommand

Possible Error and Resolution

However, if the actual data in an existing database table does exceed the composite index maximum limit during migration, the Messaging Server will show an error when it is being restarted. The error will look like this:

```
{[Microsoft][ODBC SQL Server Driver][SQL Server]Operation failed. The index
entry of length 906 bytes for the index 'rCIM_SoftwareFeatureElementsI'
exceeds the maximum length of 900 bytes.}
```

This error identifies the Inventory table whose data exceeds this maximum index value. The database error must be resolved in one of the following ways.

- If the data does not need to be retained, the table can be dropped from your database and will be recreated upon Messaging Server startup. Note, however, that this option does not prevent against subsequent entries or updates to the database table resulting in the composite index exceeding the maximum.
- If the data is valid and needs to be retained, the table can be converted back to a varchar datatype to resolve the error. To do this, create a script to convert the table back to varchar, run the script against your database, and then restart the Messaging Server. As an example, the following sample script was created to convert a table back to the varchar datatype. To generate this sample script, we performed a search on each of the migration scripts for the table name "rCIM_SoftwareFeatureElements".

Sample script to revert a table from the datatype nvarchar to varchar

```
DROP INDEX
[dbo].[rCIM_SoftwareFeatureElements].[rCIM_SoftwareFeatureElementsI]

GO


ALTER TABLE [dbo].[rCIM_SoftwareFeatureElements]

    ALTER COLUMN [userid] [varchar] (32) Collate  Latin1_General_CI_AS

GO


ALTER TABLE [dbo].[rCIM_SoftwareFeatureElements]
```

```
    ALTER COLUMN [wGroupComponent] [varchar] (255) Collate
Latin1_General_CI_AS

GO


ALTER TABLE [dbo].[rCIM_SoftwareFeatureElements]

    ALTER COLUMN [wNamespace] [varchar] (128) Collate  Latin1_General_CI_AS

GO


ALTER TABLE [dbo].[rCIM_SoftwareFeatureElements]

    ALTER COLUMN [wPartComponent] [varchar] (255) Collate
Latin1_General_CI_AS

GO


CREATE  UNIQUE INDEX [rCIM_SoftwareFeatureElementsI] ON
[dbo].[rCIM_SoftwareFeatureElements]([userid], [wNamespace],
[wGroupComponent], [wPartComponent]) ON [PRIMARY]

GO
```

After you run the script(s) on the database to revert the affected table(s) to a varchar datatype, restart the Messaging Server service and check the logs to validate that the module started without error.

SQL Server Migration for Unicode

There are five scripts for the conversion of a SQL Server database. They are located on the CM media at:

```
Infrastructure\extended_infrastructure\messaging_server\
migrate\SQL_Server
```

Run the scripts using the Microsoft SQL Server Enterprise Manager Query Analyzer tool.

If your SQL Server database was migrated to Unicode for Messaging Server Version 5.00, just run the two scripts: Step5_Modify_Indexes_MSSQL.sql and Step6_rWin32_Service_Update_SQLServer.sql.

- Step1_Drop_Indexes_MSSQL.sql

  The first script will drop the indexes from the standard table in the database

- Step2_Alter_Database_MSSQL.sql

  Follow these instructions before executing a script:

  a Replace RIMDB by your Inventory manager database Name.

  b Replace [NEW Collation Name] by your new desired Name for example French_CI_AS etc.

  c To run the script, the program needs to set the database into single user mode.

    You should therefore ensure that there are no open connections on the database before running the script (use the stored -- procedure SP_WHO to identify any open connections).

    You may want to use Kill command to Force logout the connections to the database.

d If you are running these commands from SQL Query Analyzer, its preferable Choose a different database eg Master and Run -- the below commands.

- Step3_Alter_Inventory_Tables_MSSQL.sql

  This script alters the Table columns to support Unicode strings with the Collate of your choice.

  Replace Latin1_General_CI_AS with the Desired Collate of your choice

- Step4_Add_Indexes_MSSQL.sql

  This script adds the indexes back to the standard tables.

- Step5_Modify_Indexes_MSSQL.sql

  This script corrects any UNIQUE indexes that were previously created as NON-UNIQUE.

- Step6_rWin32_Service_Update_SQLServer.sql

  This step modifies the rWin32_Service table to set the wDescription column length to 512.

Oracle Migration for Unicode

If your Oracle database was migrated to Unicode for Messaging Server Version 5.00, just run the last two scripts: Step4_Alter_Tables_Oracle.sql and Step5_rWin32_Service_Update_Oracle.sql.

- Step1_Drop_Indexes_Oracle.sql
- Step2_Alter_Tables_Oracle.sql
- Step3_Create_Indexes_Oracle.sql
- Step4_Alter_Tables_Oracle.sql
- Step5_rWin32_Service_Update_Oracle.sql

A Inventory Manager Database - Tables and Scripts

Presently there are 142 Inventory Manager Database tables created by default from the .sql scripts in the etc\core, etc\wbem and etc\inventory directories of the Messaging Server

Scripts added as of Messaging Server V 7.20 release:

- win32_physicalmemory.sql
- scripts for Vulnerability Management tables, see entries starting on page 35

Scripts added as of the Messaging Server V 5.0 release include:

- hpprov_biosenumeration.sql
- hpprov_biospassword.sql
- hpprov_biosstring.sql
- hpprov_biosorderedlist.sql
- win32_portablebattery
- win32_baseboard.sql
- win32_quickfixengineering.sql

Tables created with the `.sql` files included in the HPCA Messaging Server 7.20 release are listed below.

SQL Filename	**Table in Inventory Manager Database**
apps.jobparm.sql	JOBPARM
apps.jobstat.sql	JOBSTAT
apps.jobstat.sql	HJOBSTAT
apps.jobtask.sql	JOBTASK
apps.msiservices.sql	AppMSIEvent
apps.msiservices.sql	HAppMSIEvent
apps.rnpservices.sql	AppRNPEvent
apps.rnpservices.sql	HAppRNPEvent
apps.services.sql	AppEvent
apps.services.sql	HAppEvent
device.config.sql	DeviceConfig

device.config.sql	HDeviceConfig
device.errors.sql	DeviceErrors
device.errors.sql	HDeviceErrors
device.map.sql	DeviceMap
device.services.sql	DeviceServices
device.state.sql	DeviceState
device.state.sql	HDeviceState
device.status.sql	DeviceStatus
device.status.sql	HDeviceStatus
device.synopsis.sql	DeviceSynopsis
device.zrstate.sql	DeviceZRState
device.zrstates.sql	DeviceZRStates
fileaudit.sql	FileAudit
notify.sql	DeviceNotify
query.sql	Query
smbios.info.sql	SMBiosInfo
usergroup.sql	DeviceUserGroup
fileaudit.sql	FileAudit

cim_cdromdrive.sql	rCIM_CDROMDrive
cim_computersystem.sql	rCIM_ComputerSystem
cim_directory.sql	rCIM_Directory
cim_diskdrive.sql	rCIM_DiskDrive
cim_dvddrive.sql	rCIM_DVDDrive
cim_ethernetadapter.sql	rCIM_EthernetAdapter
cim_export.sql	rCIM_Export
cim_hpux_swbundles.sql	rCIM_HPUX_SwBundles

cim_idecontroller.sql	rCIM_IDEController
cim_logicaldisk.sql	rCIM_LogicalDisk
cim_logicaldiskbasedonvolume.sql	rCIM_LogicalDiskBasedOnVolume
cim_mediapresent.sql	rCIM_MediaPresent
cim_nfs.sql	rCIM_NFS
cim_operatingsystem.sql	rCIM_OperatingSystem
cim_parallelcontroller.sql	rCIM_ParallelController
cim_process.sql	rCIM_Process
cim_processor.sql	rCIM_Processor
cim_product.sql	rCIM_Product
cim_productsoftwarefeatures.sql	rCIM_ProductSoftwareFeatures
cim_residesonextent.sql	rCIM_ResidesOnExtent
cim_scsicontroller.sql	rCIM_SCSIController
cim_scsiinterface.sql	rCIM_SCSIInterface
cim_service.sql	rCIM_Service
cim_softwareelement.sql	rCIM_SoftwareElement
cim_softwarefeature.sql	rCIM_SoftwareFeature
cim_softwarefeaturesoftwareelements.sql	rCIM_SoftwareFeatureElements
cim_storagevolume.sql	rCIM_StorageVolume
cim_unixcomputersystem.sql	rCIM_UnixComputerSystem
cim_unixlocalfilesystem.sql	rCIM_UnixLocalFileSystem
cim_unixoperatingsystem.sql	rCIM_UnixOperatingSystem

hp_biosenumeration.sql	rhp_biosenumeration
hp_biosevent.sql	rhp_biosevent
hp_biosinteger.sql	rhp_biosinteger
hp_biosorderedlist.sql	rhp_biosorderedlist

hp_biospassword.sql	rhp_biospassword
hp_biossensor.sql	rhp_biossensor
hp_biosstring.sql	rhp_biosstring

nvd_downloadstatistics.sql	rNVD_DownloadStatistics
nvd_groupaccount.sql	rNVD_GroupAccount
nvd_groupmember.sql	rNVD_GroupMember
nvd_installed_apps.sql	rNVD_INSTALLED_APPS
nvd_installed_uninstall.sql	rNVD_INSTALLED_UNINSTALL
nvd_multicaststatistics.sql	rNVD_MulticastStatistics
nvd_nisgroupaccount.sql	rNVD_NISGroupAccount
nvd_nisuseraccount.sql	rNVD_NISUserAccount
nvd_pdasystem.sql	rNVD_PDASystem
nvd_product.sql	rNVD_Product
nvd_solarispatch.sql	rNVD_SolarisPatch
nvd_useraccount.sql	rNVD_UserAccount
nvd_wbemstatus.sql	rNVD_WBEMStatus

registry.sql	rRegistry
wifi_networkadapter.sql	rWiFi_NetworkAdapter

win32_baseboard.sql	rwin32_baseboard
win32_bios.sql	rWin32_BIOS
win32_bootconfiguration.sql	rWin32_BootConf
win32_bus.sql	rWin32_Bus
win32_cachememory.sql	rWin32_CacheMemory
win32_cdromdrive.sql	rWin32_CDROMDrive

win32_computersystem.sql	rWin32_ComputerSystem
win32_computersystemproduct.sql	rWin32_ComputerSystemProduct
win32_desktop.sql	rWin32_Desktop
win32_desktopmonitor.sql	rWin32_DesktopMonitor
win32_devicememoryaddress.sql	rWin32_DeviceMemoryAddress
win32_diskdrive.sql	rWin32_DiskDrive
win32_diskpartition.sql	rWin32_DiskPartition
win32_displayconfiguration.sql	rWin32_DisplayConf
win32_displaycontrollerconfiguration.sql	rWin32_DisplayControllerConf
win32_dmachannel.sql	rWin32_DMAChannel
win32_environment.sql	rWin32_Environment
win32_floppycontroller.sql	rWin32_FloppyController
win32_floppydrive.sql	rWin32_FloppyDrive
win32_group.sql	rWin32_Group
win32_idecontroller.sql	rWin32_IDEController
win32_irqresource.sql	rWin32_IRQResource
win32_keyboard.sql	rWin32_Keyboard
win32_loadordergroup.sql	rWin32_LoadOrderGroup
win32_logicaldisk.sql	rWin32_LogicalDisk
win32_logicalmemoryconfiguration.sql	rWin32_LogicalMemoryConf
win32_logicalprogramgroup.sql	rWin32_LogicalProgramGroup
win32_memoryarray.sql	rWin32_MemoryArray
win32_memorydevice.sql	rWin32_MemoryDevice
win32_motherboarddevice.sql	rWin32_MotherboardDevice
win32_networkadapter.sql	rWin32_NetworkAdapter
win32_networkadapterconfiguration.sql	rWin32_NetworkAdapterConf
win32_networkconnection.sql	rWin32_NetworkConnection

win32_networkloginprofile.sql	rWin32_NetworkLoginProfile
win32_operatingsystem.sql	rWin32_OperatingSystem
win32_pagefile.sql	rWin32_PageFile
win32_pagefilesetting.sql	rWin32_PageFileSetting
win32_pagefileusage.sql	rWin32_PageFileUsage
win32_parallelport.sql	rWin32_ParallelPort
win32_physicalmemory.sql	rWin32_PhysicalMemory
win32_pnpentity.sql	rWin32_PnPEntity
win32_pointingdevice.sql	rWin32_PointingDevice
win32_portablebattery.sql	rwin32_portablebattery
win32_portresource.sql	rWin32_PortResource
win32_printer.sql	rWin32_Printer
win32_process.sql	rWin32_Process
win32_processor.sql	rWin32_Processor
win32_product.sql	rWin32_Product
win32_quickfixengineering.sql	rwin32_quickfixengineering
win32_serialport.sql	rWin32_SerialPort
win32_service.sql	rWin32_Service
win32_share.sql	rWin32_Share
win32_softwareelement.sql	rWin32_SoftwareElement
win32_softwarefeature.sql	rWin32_SoftwareFeature
win32_sounddevice.sql	rWin32_SoundDevice
win32_startupcommand.sql	rWin32_StartupCommand
win32_systemdriver.sql	rWin32_SystemDriver
win32_systemenclosure.sql	rWin32_SystemEnclosure
win32_timezone.sql	rWin32_TimeZone
win32_usbcontroller.sql	rWin32_USBController

win32_useraccount.sql	rWin32_UserAccount
win32_videocontroller.sql	rWin32_VideoController

SQL Database tables added for Vulnerability Management

SQL Filename	SQL Database Table	Description
`acqprocess.sql`	**`VM_ACQUISITION_PROCESS`**	Contains the results of the update process.
`aquisitions.sql`	**`VM_ACQUISITION_FILE`**	Contains the details of the data which was acquired for a platform
`cve.sql`	**`VM_CVE`**	Contains the CVEs that are current in the system.
`cve_url.sql`	**`VM_CVE_URL`**	Contains the links that are delivered with each CVE.
`devicescan.sql`	**`VM_DEVICE_SCAN`**	Holds the summary of the scan performed on the device.
`devicevulnerability.sql`	**`VM_DEVICE_VULNERABILITY`**	Contains the actual vulnerabilities reported by the device.
`oval.sql`	**`VM_OVAL`**	Contains the current OVAL definitions being tested.
`ovalplatform.sql`	**`VM_OVAL_PLATFORM`**	This is a link table between VM OVAL and VM AFFECTED PLATFORM.
`ovalsoftware.sql`	**`VM_OVAL_SOFTWARE`**	This is a link table between VM OVAL and VM AFFECTED SOFTWARE.
`platform.sql`	**`VM_AFFECTED_PLATFORM`**	Lists the software and version that are affected by the OVAL.
`software.sql`	**`VM_AFFECTED_SOFTWARE`**	Lists the platform and version that are affected by the OVAL.
`severityhist.sql`	**`VM_SEVERITY_HISTORY`**	Contains the outcomes of all vulnerability scans performed over the last year, including the number of managed client devices in each severity category at the time of the historical data rollup.