CMS Installation and Administration Guide

Release 3.0

8202009

5100-000756-000

Contents

About this Guide

The *CMS Installation and Administration Guide* provides users with the information required to install, configure, and maintain the 2Wire Component Management System (CMS).

The following information is included:

- Introducing CMS Components. Provides an overview of CMS and its components.
- Preparing for Installation. Provides tasks that should be performed and information that should be gathered before installation.
- Installing CMS. Provides details about installing CMS, verifying installation, and starting CMS.
- Configuring Additional Functionality. Lists additional CMS functionality that can be configured by 2Wire Professional Services.
- Administering CMS. Provides common administration tasks that are performed while maintaining the CMS system.
- CMS System Errors. Lists common error messages, their descriptions, and possible resolutions.
- Available MBeans. Lists the MBeans that can be monitored using Java Management Extensions (JMX).

Audience

Users of this guide should have knowledge of the following:

- The network in which CMS will be installed, including:
  - Domain Name System (DNS)
  - Firewalls
  - Load balancers
  - Multicast configuration
  - Switches
- The Red Hat Enterprise Linux operating system
- Oracle databases

Style Conventions

The following style conventions are used in this guide:

Note Notes contain incidental information about the subject. In this guide, they are used to provide additional information about the product and to call attention to exceptions.

Typographical Conventions

Convention	Used For
Blue Text	Cross references
Bold	Interface elements that are clicked or selected
Italic	Emphasis, book titles, variables, list terms
`Monospace`	Command syntax and code
`Monospace Italic`	Variables within command syntax and code

Related Documents

In addition to this guide, the CMS documentation library includes:

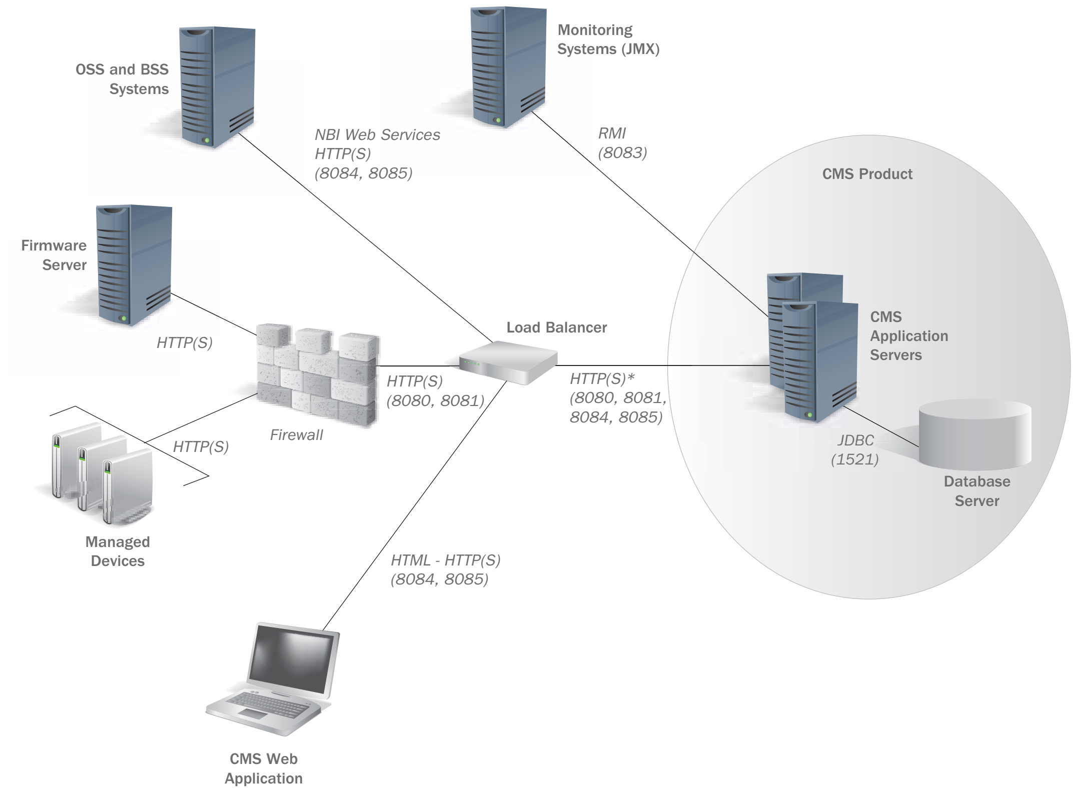
- *CMS API Reference*. Lists details about the CMS Application Programmer Interface (API) and its web services, which enable CMS to integrate with external Business Support Systems (BSS) and Operations Support Systems (OSS).
- *CMS Management Console Help*. Provides detailed instructions for working with the CMS Management Console web application.
- *CMS Overview*. Provides an introduction to CMS, including details about product functionality and concepts.
- *CMS Release Notes*. Specifies system requirements, known issues, and resolved issues for the current release.

CHAPTER 1

Introducing CMS Components

The 2Wire Component Management System (CMS) enables telecommunications service providers to manage subscriber devices.

The following deployment diagram illustrates how the CMS components and external systems are connected and which protocols and ports are used for communication in a typical installation:



Introducing the Application Server

The application server runs the CMS software and hosts the CMS Management Console. A CMS installation can have more than one application server, depending on factors such as the number of managed devices.

You are responsible for ensuring that the application server meets the minimum system requirements defined in the *CMS Release Notes*. During installation, you must install CMS software on the application server.

Using a Load Balancer

When multiple application servers are used, a load balancer equally distributes incoming service requests to the application servers. A load balancer is optional to the installation, but recommended.

During installation, you need to install a load balancer in front of the application servers and configure the load balancer to support sticky sessions to ensure that the same server processes messages related to the same session.

Using a Firewall

To protect the CMS system from external systems, a firewall may be placed between the load balancer and the devices managed by CMS. A firewall is optional to the installation.

During installation, you need to open the Secure Sockets Layer (SSL) port on the firewall to ensure that CMS traffic can pass through the firewall.

Introducing the Database Server

The database server hosts the CMS online transaction processing (OLTP) database and the optional bulk database. The CMS database stores parameters and attributes related to the devices managed by CMS, provides an audit trail of user activities, and keeps track of interactions between CMS and the devices it manages. The bulk database stores data gathered during bulk data collection. Refer to Gathering Bulk Data on page 3.

You are responsible for ensuring that the database server meets the minimum system requirements defined in the *CMS Release Notes*. You must also install the CMS database schema and bulk database on the database server.

Introducing the Firmware Server

The firmware server hosts the firmware that can be installed on devices. Each device downloads new firmware versions from the firmware server as appropriate. A CMS installation can have more than one firmware server, depending on factors such as the number of devices.

You are responsible for ensuring that the firmware server is accessible from devices and can handle the load associated with upgrading large numbers of devices simultaneously. In addition, you must maintain the correct firmware versions on the firmware server.

Introducing the CMS Management Console

The CMS Management Console is a web application that enables users to administer CMS, organize the devices managed by CMS, and resolve subscriber issues. The CMS Management Console is hosted on the application server and is available from the following location:

https://*instance*:*port*/dmc-admin

where *instance* is the IP address or host name of the application server (if you are using one application server) or load balancer (if you are using multiple application servers and a load balancer) and *port* is the internal SSL port for the application server or load balancer.

Integrating with External Monitoring Systems

You can integrate CMS with external monitoring systems using Java Management Extensions (JMX). This enables you to monitor and optimize CMS performance. Refer to Monitoring CMS with JMX on page 23.

Integrating with OSS and BSS Systems

Using web services associated with the CMS Northbound Interface, you can integrate CMS with Operational Support System (OSS) and Business Support System (BSS) applications. The data from these systems can be used to carry out or propagate business operations initiated in the external systems, such as account management or service provisioning. Refer to the *CMS API Reference* or contact 2Wire Professional Services for details about this integration.

Gathering Bulk Data

Bulk data collection enables you to regularly collect and store large numbers of device parameters, such as TR-069 and TR-98 parameters. CMS retrieves these parameters each time the devices in a specified group issue periodic inform messages. CMS stores this data in a separate bulk database that is configured during CMS installation. Refer to Preparing for Bulk Data Collection on page 7.

Communicating within the CMS Network

Within the CMS network, many protocols are used for communication.

- Hypertext Transfer Protocol Secure (HTTPS) is used to secure the following:
  - Traffic between the devices and CMS

    Note HTTPS can be discontinued at the load balancer and replaced by Hypertext Transfer Protocol (HTTP).

  - Information in the CMS Management Console
  - Northbound interface communication between CMS and external OSS and BSS systems

While use of HTTP is also supported for all of these communications, it is not recommended. By default, the following ports are used:

Communication Type	HTTPS Port	HTTP Port
External Web Application Used to communicate with devices	8081	8080
Internal Web Application Used to communicate with the Northbound Interface (NBI) and CMS Management Console	8085	8084

- Java Database Connectivity (JDBC) is used by the application servers to communicate with the database server. The JDBC port is set to the same value as the default Oracle client port, which is 1521.
- Java Remote Method Invocation (RMI) is used for communication between the application server and the optional JMX monitoring systems. By default, port 8083 is used.

Note You can configure these ports in the .config.servers file. Refer to Defining Values for the .config.servers File on page 10.

CHAPTER 2

Preparing for Installation

Review this chapter to ensure that your network, servers, and database are ready for installation.

Preparing the Servers and Network

You must prepare the servers and network in the following ways:

- *Define the number of application servers required.* The number of application servers varies by installation. Contact 2Wire Professional Services to calculate the number of application servers you require based on factors such as the number of devices to be managed and the amount of Northbound Interface (NBI) traffic.
- *Ensure that system requirements are met*. Refer to the *CMS Release Notes* for system requirements.
- *Identify IP addresses for the application servers and load balancer*. The IP addresses for the application servers and load balancer should be on the same subnet.
- *Configure Domain Name System (DNS)*. You must configure DNS so that CMS servers can resolve the host names of one another.
- *Synchronize the time on your servers*. The time on the application servers and database server must be synchronized. It is recommended that you use Network Time Protocol (NTP).
- *Configure firewalls to work with CMS*. While it is not recommended that you use a firewall except between the application servers and external sites, the following table lists the modifications that must be made to firewalls between application servers and other locations within the network so that CMS can run correctly.

For Firewalls Between	Actions To Be Taken
Application servers	Allow multicast traffic
Database server and application servers	Open the Oracle Transparent Network Substrate (TNS) Listener port
External sites and application servers	Open the Hypertext Transfer Protocol Secure (HTTPS) internal port
External system integrated with CMS and application servers	Open the Hypertext Transfer Protocol (HTTP) or HTTPS internal port

- *Configure multicast and determine the multicast address and port*. Application servers use multicast to share information. All application servers must have the same multicast address and port. It is important to use a multicast address that is unique to your application servers within any broadcast domain.

  The default multicast address and port are 239.255.2.4 and 6156. If you want to change these defaults, you must set the new information in the installation.properties file during installation.

- *Disable Internet Group Management Protocol (IGMP) snooping on switches*. If IGMP snooping is enabled, multicast messaging will not work, which prevents application servers from communicating correctly.
- *Configure sticky sessions on the load balancer*. The load balancer must be configured to support sticky sessions, which ensure that the same server processes messages related to the same session.
- *Obtain root access*. In order to install CMS and start and stop the application server as a service, you must have `root` access.
- *Allot space on application servers for CMS files*. During installation, you need to install the CMS files on each application server.
- *Obtain a CMS server certificate signed by a trusted CA for SSL communication*. To ensure that Secure Sockets Layer (SSL) communication between devices and CMS is secure, you must have a certificate from a trusted Certificate Authority (CA), such as Verisign. In addition, you must obtain the certificate used to sign the server's certificate, to establish the certificate chain to the trusted root CA.
- *Acquire the password for the database system user*. The password for the database system user is required to install the CMS database schema and related components. During installation, you need to enter the user name and password for this user in the database installation properties file.
- *Obtain access to computer with sqlplus*. You must run database installation preparation steps on a computer with a sqlplus instance that can access the database. In addition, you must have access to a user account that can access that sqlplus instance.
- *Create directory for authentication data feed*. If you will be using the manufacturing feed to authenticate devices, you must establish a directory on the database server under which incoming device credential feeds are written in preparation for export. If you are using a RAC database cluster, the path must be a shared file system location accessible by all RAC instances. During installation, you must set the absolute path to this directory in the database installation properties file.

Configuring Firmware Servers

Devices download firmware upgrades from firmware servers that you must configure and maintain. The firmware servers must be able to handle your upgrade traffic, which can be calculated using factors such as the number of devices and the size of the upgrade files. Contact 2Wire Professional Services for assistance with this calculation.

It is important that you keep your firmware servers synchronized. Refer to the documentation for your content distribution network (CDN) for configuration recommendations.

Note The firmware servers and CDN are not part of CMS. They are the sole responsibility of the service provider.

Organizing Firmware on the Firmware Server

It is your responsibility to obtain all relevant firmware upgrade files from your device vendors. Extract these zip files into the correct locations on the firmware servers.

2Wire and non-2Wire firmware images should be stored separately. Assuming your CDN has a URL that maps to a directory on the file system, unpack 2Wire firmware images in the following directory structure:

```
OUI/firmwareVersion/firmwarePackageFilename
```

where *OUI* is 00D09E and *firmwareVersion* is the version of the firmware from the imported metadata XML, or for devices with dependent devices (such as an iNID and its remote bridges), the device's firmware version, a hyphen, and the dependent device's firmware version. You must set this firmware location when you add a firmware entry in the CMS Management Console.

Place non-2Wire images in the following directory structure:

```
OUI/firmwareVersion/location
```

where *firmwareVersion* is the version of firmware and the *location* matches the location of the firmware file.

Defining the URL for Devices

After you have configured your firmware servers on your network, you must configure the URL used to access the firmware. This URL has two parts, which are joined automatically by the server:

- The first part of the URL is the firmware base URL, which you specify for each node of the policy hierarchy in the CMS Management Console.
- The second part of the URL is the location of the firmware on the firmware server, which you specify when adding firmware entries in the CMS Management Console.

Ensuring Device Access

You are responsible for ensuring that devices can access the firmware servers and can download files.

Preparing for Bulk Data Collection

CMS enables you to regularly collect parameter data from groups of devices in order to evaluate that data in external systems. The bulk data collection database is installed with the production database. You must perform the following tasks:

- *Define the bulk data collection properties required for bulk database installation*. Refer to Defining Values for the database_env_vars.sh File on page 11.
- *Ensure that there is enough space for the bulk database*. Contact 2Wire Professional Services to calculate the space required for the bulk database.
- *Define the amount of time that the data collected is retained*. You need to set this retention time in an Oracle job that regularly purges old data from the database. Refer to Configuring Oracle Jobs on page 16.
- *Define the parameters to be collected*. In the CMS Management Console, you need to define the parameters to be collected. Refer to the Bulk Data Collection section of Help.

After the data has been collected, you can report on the data in the bulk database directly using your own reporting solution, such as Crystal Reports. Alternately, you can extract the data collected to an external system where you can analyze the data as needed.

Defining Values for CMS Configuration Files

The CMS configuration files define the global configuration, global behavior modification, and server instance configuration. In addition, a database configuration file defines the information related to database schema installation.

The following tables define the configuration values that you need to use during installation.

Defining Values for the installation.properties File

The installation.properties file contains properties related to the entire CMS installation. During installation, you will configure this file and copy it to each application server.

The following properties are available for you to define:

Properties in the installation.properties File

Property	Description
JDBC-Related Configuration	
jdbc.url	The OLTP connection URL. Default: jdbc:oracle:thin:@oltp_oracle_host:*oltp_oracle_port/ oltp_oracle_service* Note: This property must be configured.
jdbc.username	The OLTP user name. This value must match the value of the OLTP_USERNAME property in the database configuration file. Note: This property must be configured.
jdbcBulk.url	The bulk database connection URL. Default: jdbc:oracle:thin:@oltp_oracle_host:*oltp_oracle_port/ oltp_oracle_service*
jdbcBulk.username	The bulk database user name. This value must match the value of the BULK_DBA_USERNAME property in the database configuration file.
jdbc.initialSize	The initial number of database connections allocated on each application server.
jdbc.maxActive	The maximum number of connections allocated to the pool on each application server. Note that the database server's max processes configuration may need to be adjusted to support this value multiplied by the number of application servers.
jdbc.maxIdle	The maximum number of active connections that can remain idle in an application server's connection pool without extra connections being released. Negative value for no limit.
jdbc.minIdle	The minimum number of active connections that can remain idle in the pool without extra connections being created, or zero to create none.
jdbc.maxWait	The maximum time, in milliseconds, that the connection pool will block the application server when attempting to allocate an active connection before an exception is thrown.
Search Results Configuration	
search.maxResultCount	The maximum number of devices returned on a search query from the CMS Management Console. Default: 2000
ActiveMQ JMS Configuration	
jms.activemq.multicast.ipaddress	The IP address of the multicast server. This address must be unique to your application servers within your domain. Default: 239.255.2.4

Properties in the installation.properties File

Property	Description
jms.activemq.multicast.port	The port number of the multicast server. Default: 6156
jms.activemq.multicast.ttl	The time-to-live setting, which defines whether traffic can cross subnets. Default: 1 (Traffic cannot cross subnets.)
CWMP Message Content Validation	
cwmpws.validateResponse	Whether to validate outbound CWMP message content that is generated by the server. This property impacts performance when it is set to true. Default: false
Web Service Request and Response Validation	
nbiws.validateRequest	Whether web service requests (other than those from Twowire Management) will be validated against the schema. This property impacts performance when it is set to true. Default: false
nbiws.validateResponse	Whether web service responses (other than those from Twowire Management) will be validated against the schema. This property impacts performance when it is set to true. Default: false
nbiws.ext.2wire.validateRequest	Whether the requests from the Twowire Management web services will be validated against the schema. This property impacts performance when it is set to true. Default: false
nbiws.ext.2wire.validateResponse	Whether the responses from the Twowire Management web services will be validated against the schema. This property impacts performance when it is set to true. Default: false
Time Skew for Database and Server	
max.server.database.time.skew	The maximum number of minutes out of sync that the time set on the database server and the application server can be set. Default: 60
CWMP Connection Request Service Configuration	
cwmpConnectionRequestService.threadPoolSize	The size of the thread pool for the CWMP connection request service. Default: 5
cwmpConnectionRequestService.retryPolicy.maxAttempts	The number of attempts that the CWMP connection request service will make to connect. Default: 3
cwmpConnectionRequestService.retryPolicy.perAttemptBackoffMillis	The number of milliseconds that the CWMP connection request service will spend attempting to connect. Default: 5000
cwmpConnectionRequestService.retryPolicy.retryImmediatelyAfterFirstAttempt	Whether to try to connect again immediately after the first attempt fails. Default: true
JMS Topic for Entity-Based Audit Event Messages	
audit.entity.topicName	The name of the JMS topic used for entity-based audit event messages published to the cluster. Default: auditEventTopic
List of Non-Unique Subscriber Identifiers Not Used for Automatic Association	
workflow.action.autoprov.nonUniqueIdentifierValues	A comma-separated list of non-unique subscriber identifier values for which no automatic association to a subscriber should be attempted.
JMS Heartbeat	
jms.heartbeat.interval	The number of minutes that the host will wait to send its heartbeat messages to the topic. Default: 100

Properties in the installation.properties File

Property	Description
Network View Timeout	
network.view.timeout	The number of minutes the CMS Management Console will attempt to retrieve network view data from the device. Default: 2
Bulk Data Persistence	
dc.maxInsertsPerTx	The number of rows that will be inserted into the bulk data table per transaction. This limits the amount of rollback consumed by bulk data. Default: 10000
dc.threadPool.coreSize	The number of threads to keep running for bulk data inserts. Default: 10
dc.threadPool.maxSize	The maximum number of threads by which the thread pool can grow to service bulk data inserts. Default: 50
dc.threadPool.workQueueSize	The size of the queue used to hold pending bulk data inserts that are not being processed by an active thread. This is essentially overflow. Default: 2000
dc.threadPool.keepAliveTime	The length of time (in seconds) that a thread that is created above the coreSize will remain available prior to being shut down. Default: 600
dc.threadPool.shutdownTime	The length of time (in seconds) to await termination of in-process bulk inserts on server shutdown. Default: 100

Defining Values for the password.properties File

The password.properties file contains passwords related to the entire CMS installation. During installation, you will configure this file and copy it to the application servers.

Note The password.properties file contains sensitive parameters that must be stored in clear text. It is very important to restrict read access only to the UNIX user configured for CMS, which is the user who runs the server. Do not provide access to any other users.

Define the following properties:

Properties in the password.properties File

Property	Description
jdbc.password	The OLTP password. This value must match the value of the OLTP_PASSWD property in the database configuration file.
jdbcBulk.password	The bulk database password. This value must match the value of the BULK_DBA_PASSWD property in the database configuration file.

Defining Values for the .config.servers File

The .config.servers file lists instance-specific configuration information, such as alternate HTTP or HTTPS ports for individual server instances, that is required when running multiple instances on a single server. During installation, you will configure this file and copy it to the application servers.

Define the following properties:

Properties in the .config.servers File

Property	Description
server1.id	The identifier for the server, which must be unique for each server only if multiple server instances are running on the same server or multiple servers' logs are being aggregated to the same location. Default: server1
server1.type	This value is dmc.
server1.Dsvcext.port.http	The HTTP port for the server to communicate externally with devices. Default: 8080
server1.Dsvcint.port.http	The HTTP port for the server to communicate internally with the Northbound Interface (NBI) or CMS Management Console. Default: svcext.port.http+4 (8084)
server1.Dsvcext.port.https	The HTTPS port for the server to communicate externally with devices. Default: svcext.port.http+1 (8081)
server1.Dsvcint.port.https	The HTTPS port for the server to communicate internally with the Northbound Interface or CMS Management Console. Default: svcext.port.http+5 (8085)
server1.Dport.shutdown	The shutdown port for the server, which is part of the Tomcat servlet container implementation. Default: svcext.port.http+2 (8082)
server1.Dcom.sun.management.jmxremote.port	The JMX port for the server. Default: svcext.port.http+3 (8083)

Defining Values for the database_env_vars.sh File

The database_env_var.sh file contains information that can be configured for the database schema installation.

Define the following properties:

Database Installation Properties

Property	Description
SQLPLUS	Absolute path to the Oracle sqlplus binary, which is used to invoke the SQL scripts that install the system's database. This binary must be installed on the computer on which you are running the database installation script.
OLTP_USERNAME	User name to be created for the server. This must match the value used for the jdbc.username server configuration parameter in the installation.properties file.
OLTP_PASSWD	Password for the OLTP_USERNAME user. This must match the value used for the jdbc.password server configuration parameter in password.properties file.
OLTP_DBA_USERNAME	Database user with rights to create all the tablespaces, user, and schema elements. Note: You will log on as this user to run the installation script. Default: system
OLTP_DBA_PASSWD	Password for the DBA_USERNAME user.
OLTP_ORACLE_HOST	Host name of the database server. This is the value returned from the hostname command.
OLTP_ORACLE_PORT	Oracle client (TNS) port of the database server to which to connect through sqlplus. Default: 1521

Database Installation Properties

Property	Description
OLTP_ORACLE_SERVICE	Service name of the database server instance.
OLTP_TS_NAME	OLTP default tablespace name.
OLTP_TS_FILE	OLTP default tablespace file location.
OLTP_INDEX_TS_NAME	OLTP index tablespace name.
OLTP_INDEX_TS_FILE	OLTP index tablespace file location.
OLTP_HIST_TS_NAME	OLTP historical data tablespace name.
OLTP_HIST_TS_FILE	OLTP historical data tablespace file location.
OLTP_HIST_INDEX_TS_NAME	OLTP historical index tablespace name.
OLTP_HIST_INDEX_TS_FILE	OLTP historical index tablespace file location.
OLTP_AUTH_TS_NAME	OLTP device authentication data tablespace name.
OLTP_AUTH_TS_FILE	OLTP device authentication data tablespace file location.
AUTH_DATA_FEED_DIR	Absolute path of the directory on the database server under which incoming device credential feeds are written in preparation for import. You created this directory during installation preparation.
BULK_USERNAME	User name to be created for bulk data storage.
BULK_PASSWD	Password for the BULK_USERNAME user.
BULK_DBA_USERNAME	Database user with rights to create the bulk data storage user, tablespaces, and schema elements.
BULK_DBA_PASSWD	Password for the BULK_DBA_USERNAME user.
BULK_ORACLE_HOST	Host name of the bulk database server.
BULK_ORACLE_PORT	Oracle client (TNS) port of the bulk database server.
BULK_ORACLE_SERVICE	Service name of the bulk database server.
BULK_TS_NAME	Bulk database default tablespace name.
BULK_TS_FILE	Bulk database default tablespace file location.
BULK_INDEX_TS_NAME	Bulk database index tablespace name.
BULK_INDEX_TS_FILE	Bulk database index tablespace file location.

CHAPTER 3

Installing CMS

Before installing CMS, ensure that all installation requirements are met. Refer to Preparing for Installation on page 5. Installation includes the following steps:

1. Installing the RPM
2. Configuring Files on the Application Servers
3. Installing and Configuring the Database Schema
4. Authenticating with Certificates
5. Starting and Stopping the Application Servers
6. Verifying Installation

Installing the RPM

CMS is installed using an RPM file. This file installs CMS as a service on the application server, creates the `dmc` user and group used to run CMS, and assigns ownership of the files to that user. Installing the RPM requires the `root` user access that you obtained during installation preparation. Refer to Preparing the Servers and Network on page 5.

To install the RPM and CMS files:

1. Log on to the application server as `root`.
2. Copy the RPM file to the application server.
3. Use the RPM file to install the CMS files. Enter:

   ```
   rpm -i rpmfile
   ```

   where *rpmfile* is the name of the RPM file.

The RPM installs the following file structure:

- /etc/init.d/dmcserver. Installed service control script
- /opt/2wire/dmc/bin. Server control scripts
- /opt/2wire/dmc/certs. Server keystore
- /opt/2wire/dmc/conf. Server configuration files
- /opt/2wire/dmc/container. Tomcat application container
  - conf/server.xml. Container configuration
  - conf/webapps. Product web applications
- /opt/2wire/dmc/db. Database scripts
- /opt/2wire/dmc/ext. 2Wire Professional Services extension area
- /opt/2wire/dmc/jdk. Java Virtual Machine

- /opt/2wire/dmc/lib. Product libraries
- /opt/2wire/dmc/logs. Server log file location
- /opt/2wire/dmc/scripts. Service control scripts
- /var/run/dmc. PID file location for service control

Note Unless otherwise specified, the remainder of the installation and configuration steps must be executed as a user that has permission to read and modify the files owned by the `dmc` user or group.

Configuring Files on the Application Servers

You must modify a configuration file to increase the number of open files allowed. You must also configure the CMS properties files on each application server and ensure that the configuration files on all application servers are synchronized.

Setting the Open Files Limit in limits.conf

You must increase the open file limits in limits.conf to ensure that CMS runs smoothly under heavy use.

To increase the open file limits:

1. Log on to the application server as `root`.
2. In a text editor, open the /etc/security/limits.conf file.
3. Add the following lines to the file:

   ```
   dmc hard nofile 10240
   dmc soft nofile 10240
   ```
4. Save and close the file.
5. Repeat these steps on each application server.

Editing the installation.properties File

The installation.properties file contains general properties related to your CMS installation. For information about these properties, see Defining Values for the installation.properties File on page 8.

To edit a property in the installation.properties file:

1. Log on a user that has permission to read and modify the files owned by the `dmc` user or group.
2. In a text editor, open the *CMS_install_dir*/dmc/conf/installation.properties file.
3. Modify one or more properties.
4. Save and close the file.

Editing the password.properties File

The password.properties file contains all the passwords used in the CMS installation. For information about these properties, see Defining Values for the password.properties File on page 10.

To edit a property in the password.properties file:

1. In a text editor, open the *CMS_install_dir*/dmc/conf/password.properties file.
2. Modify one or more properties.
3. Save and close the file.

Editing the .config.servers File

The .config.servers file lists instance-specific configuration information. For information about these properties, see Defining Values for the .config.servers File on page 10.

To edit a property in the .config.servers file:

1. In a text editor, open the *CMS_install_dir*/dmc/conf/.config.servers file.
2. Modify one or more properties.
3. Save and close the file.

Copying the CMS Files to the Application Servers

After you have completed the configuration on one application server, copy the CMS files in the *CMS_install_dir* directory to the same location on the other application servers.

Installing and Configuring the Database Schema

The database schema installation installs the database schema and default Oracle jobs, which enable the automatic purgeof old data in the database.

You must configure the database schema installation script before installation and the oracle jobs after installation.

Editing the database_env_vars.sh File

The database schema installation requires the database_env_var.sh file, which contains information that can be configured for the installation. For information about these properties, see Defining Values for the password.properties File on page 10.

Note You must perform the following procedure on a computer with a sqlplus instance that has access to the database. If this computer is not one of the application servers, you must copy the *CMS_install_dir*/dmc/db directory from an application server to the computer.

To edit the database_env_var.sh file:

1. Log in as a user who can access the sqlplus instance.
2. Open the *CMS_install_dir*/dmc/db/database_env_vars.sh file in a text editor.

3. Set the properties.
4. Save and close the file.

Touching the Authorization Files

To touch the authorization files in the `auth_dat_feed_dir` directory, a location you enter in the database_env_var.sh file:

1. On the database server, log in as the Oracle database user.
2. Open the `auth_dat_feed_dir` directory. At the command prompt, enter:

```
touch modelmanufacturer.txt

touch auth_data_current_1.txt

touch auth_data_complete_1.txt
```

Installing the Database Schema

To install the database schema:

1. On the server with a sqlplus instance, log in as a user who can access the sqlplus instance.
2. Run the installation script. At the command prompt, enter:

```
CMS_install_dir/dmc/db/create_db.sh
```

3. To verify that installation was successful, verify that you can access the Oracle jobs used to clean up the database. Refer to Configuring Oracle Jobs on page 16.

Note Because the database installation properties script contains passwords related to the database, it is very important that you delete the passwords from the file or delete the file itself after installation.

Configuring Oracle Jobs

Some tables in the OLTP database require periodic maintenance to purge old data and perform other routine maintenance in order to maintain the level of performance of the system. This process is performed automatically by Oracle jobs in the *CMS_install_dir*/dmc/db directory. Within these files:

- The job_name is the name of the job.
- The argument_position is the position of the program argument being set.
- The argument_value is the value set for the program argument.

The create_maint_jobs.sql file contains jobs to maintain the OLTP database. By default, these jobs purge old interval partitions at the intervals specified.

Oracle Jobs in create_maint_jobs.sql

Job Name	Description	Runs	Data Retained
PURGE_AUDIT_ENTRY_DATA	Purges old interval partitions on the audit_entry table	Monthly	Three months
PURGE_FIRMWARE_UPGRADE_HIST	Purges old interval partitions on the firmware_upgrade_hist table	Monthly	Three months
PURGE_SCHED_WF_PROC_DEVICE	Purges old interval partitions on the scheduled_workflow_proc_device table	Monthly	Three months
PURGE_TERM_DEVICE_WORKFLOW	Purges old interval partitions on the term_device_workflow table	Monthly	Three months
PURGE_TERM_DEVICE_WFLOW_VAL	Purges old interval partitions on the term_device_wflow_val table	Monthly	Three months
PURGE_TRACE_CONVERSATIONS	Purges old interval partitions on the trace_conversations table	Monthly	Three months
PURGE_TRACE_ELEMENT	Purges old interval partitions on the trace_element table	Monthly	Three months
PURGE_TWOWIRE_PKG_UPGRADE_HIST	Purges old interval partitions on the twowire_pkg_upgrade_hist table	Monthly	Three months
UPDATE_FIRMWARE_ERROR_STATUS	Updates firmware status for records with a status of E	Daily	N/A

In addition, the create_maint_jobs.sql file contains a job to maintain the bulk data collection table.

Oracle Job in create_maint_jobs_sql

Job Name	Description	Runs	Data Retained
PURGE_BULK_DATA	Purges old interval partitions on bulk_data table	Daily	Five days

You can configure these jobs to modify how frequently they run. For example, the PL/SQL code to purge the audit_entry table every six months instead of every three months is as follows:

```
BEGIN
DBMS_SCHEDULER.SET_JOB_ARGUMENT_VALUE('PURGE_AUDIT_ENTRY_DATA', 1, '6');
END;
/
```

Authenticating with Certificates

During installation preparation, you obtained a Secure Sockets Layer (SSL) certificate from a trusted Certificate Authority (CA). This certificate ensures that SSL communication between CMS and the devices that it manages is secure. You can choose to terminate SSL at the load balancer or at the application servers.

Terminating SSL at the Load Balancer

To terminate SSL at the load balancer, import the certificate into the load balancer. Refer to the load balancer documentation for instructions. This is recommended if the load balancer can handle the traffic because it frees the application servers from having to decrypt and encrypt inbound and outbound SSL.

Terminating SSL on the Application Servers

To terminate SSL on the application server:

1. Replace the *CMS_install_dir*/dmc/certs/dmc.key file with the generated private key that you used to request the certificate from a trusted Certificate Authority (CA).
2. Replace the dmc.crt file with the CA-signed certificate.
3. Replace the cacert.pem file with the updated certificate chain, which is used to sign the server's certificate.

Starting and Stopping the Application Servers

You can start and stop the application servers from the command line.

Starting Application Servers

To start an application server:

1. Log onto the application server as `root`.
2. Start the service. At the command prompt, enter:

   ```
   service dmcserver1 start
   ```

Stopping Application Servers

To stop an application server:

1. Log on to the application server as `root`.
2. Stop the service. At the command prompt, enter:

   ```
   service dmcserver1 stop
   ```

Verifying Installation

To verify installation was successful, start the application server using the startup script and then perform one or more of the following tasks:

- Verify that the java process for the server is running. At the command prompt, enter:

  ```
  ps p ‘cat /var/run/dmc/dmcserver1’
  ```

- The network should be listening on the following ports:
  - http clear (default 8080, 8084)
  - https (default 8081, 8085)
  - Tomcat shutdown (default 8082)
  - jmx (default 8083)

  Note The values for these ports are set in the .config.servers file.

  To verify the status of a port, enter the following at the command prompt:

  ```
  netstat -an port
  ```

- Check for a server startup message with the current date in the dmc_*instance*_stdout.log file.
- Verify that there are no errors in the dmc_*instance*.log file.
- Ensure that you can access the CMS Management Console. In a browser, enter:

  https://*app_server*:*port*

  where *app_server* is the IP address or host name of the application server (if you are using one application server) or load balancer (if you are using multiple application servers and a load balancer) and *port* is the SSL internal port that you set in the .config.servers file.

  The installation creates the following users:
  - User `super` with password `2wire`. This user has the Superuser role.
  - User `csr` with password `2wire`. This user has the Support role.
  - User `nbiws` with password `2wire`. This user has the OSS/BSS System role.

  For more information about these roles, see the Users section of *CMS Management Console Help*, which is available from within the CMS Management Console web application.

Preparing CMS for Use

After you have successfully installed CMS, it is recommended that you refer to *CMS Management Console Help* and perform the following tasks:

- *Add user accounts*. Users must have a user name and password to access CMS. In addition, each OSS and BSS system should authenticate using unique credentials to ensure that the actions of each system can be tracked.
- *Configure organizations*. Organizations and their policy hierarchies define how devices are configured.

- *Create groups of devices*. Groups define the devices affected by scheduled firmware upgrades or bulk data collection.
- *Adding firmware entries*. You configured the firmware server while preparing for installation. Refer to Configuring Firmware Servers on page 6. Now, you can configure firmware entries in the CMS Management Console so that the firmware can be installed on devices.
- *Configure bulk data collection*. You optionally configured CMS to use bulk data collection. Refer to Preparing for Bulk Data Collection on page 7. Now, you can configure the parameters that will be gathered and the groups of devices that will be affected.

CHAPTER 4

Configuring Additional Functionality

Contact 2Wire Professional Services to add the following functionality to CMS:

- *Configuring SSL mutual authentication*. SSL mutual authentication is recommended for communication between external systems and CMS. With SSL mutual authentication, the external system and the CMS server authenticate one another using X.509 certificates.
- *Authenticating devices*. Devices must be authenticated in order to interact with CMS and the recommended method is through data from the manufacturing feed. This information can be imported into the CMS database incrementally to ensure that it is kept current.
- *Providing access to the WAN Management Interface*. CMS can provide access to a WAN management interface on devices. This enables CMS users to configure a device directly. The out-of-box CMS configuration enables integration with the MDC URL used for 2Wire devices. Before this type of interface can be accessed, the device must be authenticated, typically using the manufacturing feed.
- *Integrating with LDAP*. Lightweight Directory Access Protocol (LDAP) can be configured as the authentication mechanism for application users, web services, and devices.
- *Integrating with OSS and BSS systems*. Integrating with Operations Support Systems (OSS) and Business Support Systems (BSS) systems is an integral part of sharing data throughout your organization.
- *Adding device classifiers*. Device classifiers enable users to label devices and are used to search, classify policies, and select workflows. Additional device classifiers can be added to CMS.
- *Adding custom properties*. Device properties are used to search for devices, create groups, and create expressions used in the policy hierarchy. Custom device properties can be added to CMS, and their values must be communicated to CMS users so that they can use them effectively.
- *Launching CMS Management Console from an external system*. The CMS Management Console can be launched directly from an external system. For example, users could access information about a specific device in the CMS Management Console after clicking a link in a customer relationship management system.
- *Creating subscriber identifiers*. Subscriber identifiers are properties used to identify subscribers, such as account number and phone number. Additional subscriber identifiers can be added to make it easier to search for subscribers.

CHAPTER 5

Administering CMS

After you have installed CMS, you may need to perform the tasks described in this chapter to administer and maintain the system. In addition, review the error messages and responses in the appendix CMS System Errors on page 25.

For a list of system requirements and known issues, refer to the *CMS Release Notes*.

Monitoring CMS with JMX

You can monitor CMS using Java Management Extensions (JMX). CMS provides CMS-specific MBeans as well as MBeans related to third-party applications that are part of the product infrastructure, such as the Java Virtual Machine (JVM) and Tomcat servlet container. For a list of MBeans that can be monitored, see the appendix Available MBeans on page 33.

You can access JMX information through jconsole. The default configuration of the application server enables remote JMX access to port 8083, disables SSL for such interactions, and authenticates access through the *CMS_install_dir*/dmc/conf/jmxremote.access and *CMS_install_dir*/dmc/conf/jmxremote.password files.

To connect to a server using these default values:

1. Launch a JMX client, such as $JAVA_HOME/bin/jconsole.

Note On the computer where you are launching the client, you must have the JAVA_HOME variable set to the Java installation directory.

2. Establish a remote connection to server-fqdn:8083.

The user name and password for this connection are specified in the *CMS_install_dir*/dmc/conf/jmxremote.access and *CMS_install_dir*/dmc/conf/jmxremote.password files. You can modify these default settings in the *CMS_install_dir*/dmc/conf/.config.servers file.

For more information on configuring remote JMX access, refer to http://java.sun.com/j2se/1.5.0/docs/guide/management/agent.html.

Authenticating Devices with the Manufacturing Utility

Devices must be authenticated in order to interact with CMS. During installation, 2Wire Professional Services can implement a utility to pull information from the manufacturing feed, which uniquely identifies each device. The manufacturing utility imports this data into the CMS database periodically to ensure that it is kept current.

The manufacturing feed is a set of text files that contain device manufacturing data. Each line in the manufacturing feed pertains to a single device. The line of device data contains pipe-delimited entries that correspond to the device’s serial number, software version, ID string, PC assembly number, authentication code, and default wireless ID (password). Using the utility provided with CMS, the information in this file can be used to uniquely identify each device.

The manufacturing utility converts the manufacturing feed data into a set of device credentials that can be consumed by CMS. Then, it imports this information into the CMS database. The initial load of this data contains a complete set of device credentials, which can be very large. After this data is added to the database, the utility is configured to incrementally load the current set of device credentials to keep the database updated.

Stopping the Manufacturing Utility

To stop the manufacturing utility:

- On the server where the utility is installed, enter the following at the command line:

  ```
  service mfutility stop
  ```

Starting the Manufacturing Utility

To start the manufacturing utility:

- On the server where the utility is installed, enter the following at the command line:

  ```
  service mfutility start
  ```

Note These procedures assume that the manufacturing utility is running as a service. If 2Wire Professional Services did not implement the utility as a service, contact them for information about starting and stopping the utility.

Running Oracle Jobs

The CMS installation created a set of Oracle jobs used to purge old data in the database. Refer to Configuring Oracle Jobs on page 16.

It is important that you configure and maintain these jobs, which can be accessed through Oracle Enterprise Manager.

Uninstalling CMS

Uninstalling CMS removes the CMS package but does not remove the `dmc` user and group or the database schema created during installation.

To uninstall CMS:

1. Log on to the application server as `root`.
2. Stop the service. At the command prompt, enter:

   ```
   service dmcserver1 stop
   ```
3. Uninstall the CMS package. Enter:

   ```
   rpm -e rpmfile
   ```

   where *rpmfile* is the name of the RPM file that was used to install CMS.

APPENDIX A

CMS System Errors

The following system errors may be encountered during use of CMS:

A CWMP request was received outside of a valid CWMP session

The server received a CWMP message from a device for which no active CWMP session exists. Either another server was failed over to this server or the device is not complying with the TR-069 specification. The server will send an empty response to terminate the conversation with the device.

A failure occurred executing scheduled object with ID {0} of type {1}

An error occurred during the execution of the specified scheduled job. Refer to the details of the log message to determine what, if any, intervention is required. For recurring scheduled jobs, the job will continue to execute at the next scheduled recurrence.

A previous listener was already registered (and has been replaced) for context key {0}: {1}

The server encountered a pre-existing JMS listener registered for a device context. This is an internal error. Any messages sent to the previous JMS context will be dropped, and as a result, the corresponding responses will not be returned to the caller.

A transfer complete message was received from device ID {0} specifying commandKey {1} for which the system has no record

A device sent a spurious CWMP TransferComplete message with the given command key to the server.

A value was set for a profile definition rule of NONE for parameter {0} on device ID {1}

A configuration profile was defined within a policy hierarchy for which a definition value was specified, but the definition rule was NONE. This does not negatively affect the system, but should be corrected in the policy hierarchy.

An ambiguity was detected within the device policy tree: {0}

An ambiguity was found within a device policy hierarchy. To resolve this issue, update the policy hierarchy to remove the ambiguity.

An exception occurred within CWMP conversation processing

An error occurred while processing or responding to a CWMP message sent from a device. The server will terminate the CWMP session.

An exception was thrown processing queued device workflow with ID {0} for device ID {1}

An unexpected failure occurred while processing a workflow enqueued for a device. This includes unexpected messages returned from the device (other than CWMP faults) and failures of the server to respond to the device due to internal workflow execution.

An immediate completion status was returned for a download request to device ID {0}, which specified non-inline download

The specified device returned a CWMP message indicating it had already completed a requested firmware upgrade. However, the server requested a non-inline download, meaning the device should have started a separate CWMP session to relay the result of the download status to the server. The device is not complying with the TR-069 specification.

An unexpected exception was thrown within the lock acquisition task

An error occurred while attempting to acquire global locks, such as the global scheduler. If no server owns a global lock, the functionality associated with that lock will not be performed.

To resolve this issue, verify that another server in the cluster acquired the lock. If not, use JMX to force a server to relinquish and re-acquire the global locks.

An unexpected exception was thrown within the lock refresh task

The server was unable to refresh the locks it had previously acquired. After the lock has timed out, another server in the cluster should acquire the lock.

Attempting to update remote bridge {0}'s model identifier (was UNKNOWN)

An inform was received from a device with a remote bridge for which no corresponding model identifier exists in the system. This indicates manufacturing information is missing from the database, possibly as a result of a manufacturing feed problem.

To resolve this issue, ensure that model identifiers are defined in the database for any devices with the model identifier “UNKNOWN”.

Cache {0} not found. Using default configuration to create.

The server's configuration referenced a named cache that does not exist, forcing the caching infrastructure to use the default caching configuration.

To resolve this issue, correct the server configuration and restart the servers.

Cannot replicate non-serializable cache key of type {0}: {1}

A request to replicate a cached entity was made; however, the identification of the entity was not serializable, and thus could not be sent over the JMS message bus. This is an internal error with no workaround.

Device credentials import failed

The server failed to import new device credentials. To view specific details of the failure, review the exception trace or database logs.

Device ID {0} sent a firmware Package Modify Status message for key {1} and package {2} that did not match any outstanding request

A device sent an unexpected package modify status update for the specified device. The update is ignored.

Device ID {0} sent a firmware Package Modify Status message for key {1} that did not match any outstanding request

A device sent an unexpected package modify status update for the specified device. The update is ignored.

Device ID {0} sent a firmware Package Set Modify Status for key {1} that did not match any outstanding request

A device sent an unexpected package set modify status update for the specified device. The update is ignored.

Failed to convert internal CWMP response object of type {0} to outbound representation

The server failed to generate the CWMP XML representation for a response to a device message. The server will terminate the CWMP session.

Failed to convert String representation of WAN IP address {0} to numeric form for device ID {1}

The server was unable to generate the numeric representation of the specified device's WAN IP address. As a result, WAN IP address-based expressions within the system may not evaluate the device correctly.

Failed to decode product class {1} for device with OUI {0} and serial number {2}

The specified product class sent from the device could not be decoded as UTF-8.

Failed to delete scheduled job with identifier {1} associated with scheduled object ID {0}

An internal error occurred in the scheduler when attempting to delete the specified recurring scheduled job.

Failed to deploy process definition for workflow definition: {0}

The server could not re-deploy the specified workflow definition, likely due to the workflow XML being not well-formed, non-compliant to the workflow XML schema, or referencing undefined workflow action or decision implementation classes.

To resolve this issue, modify and re-deploy the workflow.

Failed to de-serialize XML sent from device: {0}

The server was unable to parse a CWMP XML message sent from a device. The server will return a CWMP fault to the device.

Failed to generate rule text for rule base ID {0} with expression ID {1}

A device expression (such as an expression used to define a group) could not be converted into a rule for the internal rule base. This is an internal error.

To resolve this issue, either modify the object containing the expression so that the system is able to generate a corresponding rule or delete the expression.

Failed to parse or compile rule text for rule base ID {0} with expression ID {1} with rule text: {2}

A rule generated for a device expression could not be compiled by the internal rule base. This is an internal error.

To resolve this issue, either modify the object containing the expression so that the system is able to compile the corresponding rule or delete the expression.

Failed to perform post-execution bookkeeping activities for the scheduled object with ID {0} of type {1}

An error occurred while performing bookkeeping activities following execution of a scheduled job. This should only occur as a result of a failure communicating with the database or of the database to perform the requested updates. As a result, the scheduled job may not have been marked inactive (if non-recurring) and no audit entry may have been created recording the job's execution.

Failed to retrieve default value for property {0} of bean class {1} on bean-based workflow definition {2}

An error occurred while checking for the default value for a bean-based workflow parameter specified by the implementation class name. The workflow will execute, but no default value will be available.

Failed to retrieve recurring scheduled job names during scheduler shutdown

The server was unable to determine the existing recurring scheduled jobs during shutdown. This is not an issue unless the schedule was being executed at the time of shutdown. In that case, the scheduled execution will be halted, but the job will continue to execute at the next scheduled recurrence.

Failed to schedule job for scheduled object ID {0}

The server was unable to schedule the specified recurring job. The scheduled job will not be executed.

Failed to serialize outbound CWMP message: {0}

The server was unable to serialize an outbound CWMP message. The server will terminate the CWMP session and return an HTTP 500 to the device.

Failed to start recurring jobs scheduler

The server was unable to start the recurring scheduler. As a result, recurring scheduled jobs will not be executed.

Failed to un-schedule recurring job during scheduler shutdown: {0}

The server was unable to un-schedule a recurring scheduled job during shutdown. This is not an issue unless the schedule was being executed at the time of shutdown. In that case, the scheduled execution will be halted, but the job will continue to execute at the next scheduled recurrence.

Firmware upgrade for device ID {0} failed with error code {1}. Message was: {2}

A firmware upgrade failed for the specified device. The CWMP fault code and detail message are specified in the message.

Ignoring submitted device data due to pool shutdown

A bulk data database insertion request was ignored because the server is shutting down.

JMS error receiving lock service message from source {0}

A failure occurred when the lock infrastructure attempted to read the contents of a lock-related message sent from another server over the JMS message bus. This indicates a failure within the messaging infrastructure, possibly caused by I/O problems between the two servers.

JMS failure reading message

A failure occurred when the cache infrastructure attempted to read the contents of a cache-related message sent from another server over the JMS message bus. This indicates a failure within the messaging infrastructure, possibly caused by I/O problems between the two servers.

JMS failure reading message

A failure occurred when the cache infrastructure attempted to read the contents of a cache-related or schedule-related message sent from another server over the JMS message bus. This indicates a failure within the messaging infrastructure, possibly caused by I/O problems between the two servers.

JMS failure reading message from source {0}

A failure occurred when the cache infrastructure attempted to read the contents of a cache-related message sent from another server over the JMS message bus. This indicates a failure within the messaging infrastructure, possibly caused by I/O problems between the two servers.

Mandatory service property {0} was not defined for service {1} on subscriber of device ID {2}

The specified, mandatory service detail property was not defined for the given service and subscriber, which prevented the HSIA provisioning workflow from being able to execute for the specified device.

To resolve this issue, update the HSIA service details on the subscriber to contain all mandatory properties.

No install or remove packages found for upgrade for device ID {0} and firmware ID {1}

The server received a request to apply 2Wire firmware to a device, but the firmware image contained no packages to remove or install. The firmware upgrade will be ignored.

No model identifier found for dependent device

An inform was received from a dependent device for which no corresponding model identifier exists in the system. This indicates manufacturing information is missing from the database, possibly as a result of a manufacturing feed problem. Ensure model identifiers are defined in the database for any devices having the model identifier “UNKNOWN”.

No model identifier found for device

An inform was received from a device for which no corresponding model identifier exists in the system. This indicates manufacturing information is missing from the database, possibly as a result of a manufacturing feed problem.

To resolve this issue, ensure that model identifiers are defined in the database for any devices having the model identifier “UNKNOWN”.

No notification history found for parameter {0} on device ID {1}

The server modified the notification attributes of the specified parameter on the device but could not find any history of the notification within the database. This is likely the result of the device being cleared or recycled concurrently with the parameters being updated.

Queue full for insertion of device data (count: {1}). Device: {0}

A bulk data database insertion request failed because the inbound insert queue was full. This indicates that the bulk data persistence thread pool is not able to keep up with the amount of incoming bulk data insertion requests.

To resolve this issue (assuming there are no systemic problems such as database or networking issues), increase the number of threads allocated to the pool through the asyncPersistencePoolService MBean.

Received invalid provisioning code for device with serial number {0} and OUI {1}: {2}. Default organization will be applied to the device.

The server could not parse the specified provisioning code sent from the device. As a result, the device was placed into the default organization. This may have occurred because invalid provisioning codes were burned into the device or because of a problem in the plugged-in provisioning code parser.

To resolve this issue, use SQL to move devices manually from the default organization to the correct organization.

Rejecting request: Authenticated principal does not match Inform for Username {0}, OUI {1}, productClass {2}, and serial number {3}

A device sent a CWMP Inform message for which the OUI, serial number, and product class do not match the principal specified in the authentication request. The server will return an HTTP 401 in response.

Server generated schema-invalid CWMP response XML: {0}

The server generated a CWMP XML message that is not compliant with the schema.

Shutting down bulk data persistence service. Currently there are {0} items in flight. Will wait for up to {1} seconds for orderly shutdown

The server is shutting down and there are in-flight bulk data inserts awaiting persistence.

Skipping firmware upgrade for device ID {0} due to existing outstanding firmware upgrade

The server was requested to initiate a firmware upgrade for the specified device, but another firmware upgrade was already in progress for the device. The newly-requested firmware upgrade is ignored.

Skipping firmware upgrade for device ID {0} due to existing outstanding firmware upgrade

The server was requested to initiate a 2Wire firmware upgrade for the specified device, but another firmware upgrade was already in progress for the device. The newly-requested firmware upgrade is ignored.

Skipping firmware upgrade for device ID {0} remote bridge due to existing outstanding firmware upgrade

The server was requested to initiate a 2Wire firmware upgrade for the specified remote bridge, but another firmware upgrade was already in progress for the remote bridge. The newly-requested firmware upgrade is ignored.

The effective policy for the device contains errors. The error was logged previously for organization {0}.

The effective policy applicable to the device contains errors. This is usually a result of the expressions used in a service policy node's child nodes, which create ambiguities such that a device can match multiple child nodes of the service policy node. The source of the ambiguities was logged with the error message code “com.twowire.dmc.service.impl.DevicePolicyServiceImpl.policyTreeAmbiguityDetected”.

To resolve this issue, update the policy hierarchy to remove the ambiguity.

The organization {0} contains a service policy node that references the service {1}, but does not contain the service in the organization priority ordering

A service policy node in the organization's policy hierarchy refers to a service that is not included in the organization's service priority list. This will result in undefined behavior with respect to determining the effective policy to be applied to a device in the organization.

To resolve this issue, either add the service to the organization's priority list, delete the service policy node, or update the service policy node to refer to a service that does exist in the organization's priority list.

Unable to load bean class {0} for bean-based workflow definition {1}

A bean-based workflow definition deployed into the system specified an invalid class implementation. Either required extension libraries were not included in the server's classpath or an incorrect class name was specified when the workflow definition was deployed.

Unable to parse CWMP ISO date sent from device. Date text was: {0}

The server was unable to parse a CWMP ISO-formatted date sent from a device.

To resolve this issue, define an additional date format in the server configuration, specified by the “cwmpDateParserPatterns” configuration property.

Unexpected exception

An unexpected exception was caught within the system.

Unsupported scheduled workflow termination status: {0}

An unsupported workflow termination status was encountered while generating a termination summary for a scheduled workflow.

APPENDIX B

Available MBeans

The MBeans in this appendix are available to be monitored using Java Management Extensions (JMX). Refer to Monitoring CMS with JMX on page 23.

com.twowire.common.jmx/Log4j

`Log4j` is the logging mechanism used within CMS. `Log4j` is based on the idea of logger categories, which are hierarchical names in which a '.' indicates a new level of hierarchy. For example, `com.twowire.dmc.services` is a child category of `com.twowire.dmc`. Each category has an effective log level, which includes, in order of decreasing verboseness: TRACE, DEBUG, INFO, WARN, ERROR, and FATAL. The log level can be fixed for a category or it can be inherited from its parent.

The `Log4j` MBean provides dynamic means of modifying the levels of `log4j` categories. The MBean exposes the following operations:

Operation	Description
getLoggerNames()	Returns the set of category names known to the system. These are categories that have been created either because the logging configuration set their levels or because code executed that referenced the categories.
getLoggers()	Returns the set of category names known to the system, including the effective log level of the category and an indication of whether the level was inherited. For example: `com.twowire.dmc.apps.nbiws.GroupManagementEndpoint[DEBUG *]` indicates that the category `com.twowire.dmc.apps.nbiws.GroupManagementEndpoint` has the effective level of DEBUG and this level was inherited, as indicated by the asterisk. Alternately: `com.twowire.common.spring.context.TransactionApplicationListener[WARN]` indicates a category fixed at the WARN level.
setLoggerLevel()	Given a category name and level name, the method fixes the named category at the named level.

com.twowire.common.monitoring/jamonPerfSummaryLogger

The `jamonPerfSummaryLogger` MBean exposes management operations on method/API metrics gathered by the system.This includes reports on method invocation counts; min, max, average, and standard deviation of execution times; first and last execution times; and current, max, and average concurrency. The MBean exposes the following operations:

Operation	Description
reportSummary()	Causes a CSV-formatted report containing all the metrics gathered since the last reset or server startup to be written to the server's log file.
viewSummary()	Returns a CSV-formatted report containing all the metrics gathered since the last reset or server startup.
resetStats()	Resets the metrics data.

com.twowire.common.persistence.util/ dataSourceConnectionTracker,dataSourceBulkConnectionTracker

The data source connection tracker MBeans (one for the primary connection pool and one for the bulk data insertion connection pool) provide tracking support for outstanding database connections. When enabled, the trackers maintain thread information, a stack trace, and a timestamp captured at the point where the connection was obtained from the pool. This provides information useful to find cases of leaked connections or long-running connection usage. The trackers automatically log the captured information to the server's log whenever all but the last connection have been taken from the pool. The MBean exposes the following operations:

Operation	Description
isEnabled()/setEnabled()	Retrieves or sets whether connection tracking is enabled on the data source.
logActiveConnections()	Writes the summaries of active tracked connections to the server's log file.
getActiveConnectionSummary()	Returns a String summarizing the active tracked connections.
getNumActiveConnections()	Returns the current number of active tracked connections.

com.twowire.dmc.apps.cwmpws/cwmpController

The `cwmpController` MBean exposes management information about the CWMP controller, which manages CWMP conversations. The MBean exposes the following operations:

Operation	Description
isTraceCwmpMessages()/ setTraceCwmpMessages()	Retrieves or sets whether all CWMP message requests and responses are logged at the TRACE level. By default, this is disabled due to the added performance cost, which is present even if the category's logger is disabled. This is useful for small-scale debugging. For production debugging, use the trace feature provided in the CMS Management Console for per-device debugging. Note that even with CWMP message tracing enabled, in order for the CWMP messages to be logged, the log4j category `com.twowire.dmc.apps.cwmpws.CwmpController` must be set to TRACE level.

com.twowire.dmc.apps.twowire.service.impl/twowireMDCService

The `twowireMDCService` MBean exposes management information for 2Wire Management Diagnostic Console (MDC) web session support. The operations on this MBean affect the time-outs and realm used when establishing web sessions with 2Wire devices.

com.twowire.dmc.dc.impl/asyncPersistencePoolService

The `asyncPersistencePoolService` MBean provides management information for asynchronous persistence of bulk data to the bulk data source. The operations on the MBean configure the threads allocated to perform this background storage.

com.twowire.dmc.grouping/groupService

The `groupService` MBean manages the facility used to track groups and specifically provides management operations affecting the rule base used to determine device membership for groups defined by expressions. The MBean exposes the following operations:

Operation	Description
getRules()	Returns the contents of the group rules from the rule base, for groups based on expressions.
getRuleCount()	Returns the number of group rules in the rule base, for groups based on expressions.
getLastRebuild()	Returns the timestamp the group rule base was last rebuilt. The rule base is naturally rebuilt whenever the system determines that a group based on an expression has been created, modified, or deleted.
rebuildRuleBase()	Rebuilds the current server's group rule base.

com.twowire.dmc.messaging/jms.heartbeater

Each server in the system communicates with the other members of its cluster through Java Message Service (JMS). The `jms.heartbeater` MBean tracks this server's perspective of the other members of the application cluster. It exposes attributes containing the set of cluster members and when the server last received heartbeat messages from each. Note the list always contains the current server, and, therefore, always has at least one member. The individual servers are identified by IP address and external clear HTTP port, since the combination of those two variables must be unique for each server.

com.twowire.dmc.service.impl/cwmpConnectionRequestService

The `cwmpConnectionRequestService` MBean manages the thread pool used to carry out CWMP connection requests to devices. The MBean exposes the following operations:

Operation	Description
getThreadPoolSize()/ setThreadPoolSize()	Retrieves or updates the number of threads allocated to the pool. This directly constrains the number of concurrent connection requests that may be performed by a server.
getNumShutdownThreadPoolsAwaitingCleanup()	When the size of the thread pool is adjusted, the old thread pool is moved aside and replaced by a new thread pool with the new size. The old thread pool's jobs, however, are allowed to execute to completion. Afterward, the thread pool is marked as "awaiting cleanup", which occurs out-of-band. This value should be 0 or, if the thread pool size was updated, 1.
getNumEnqueuedConnectionRequests()	Returns the number of enqueued connection requests awaiting execution by the thread pool.

com.twowire.dmc.service.impl/devicePolicyService

The `devicePolicyService` MBean manages the facility used to determine the effective policies to be applied to devices, and specifically provides management operations affecting the rule base used to determine the policies that apply to a given device. The MBean exposes the following operations:

Operation	Description
getRules()	Returns the contents of the policy rules from the rule base.
getRuleCount()	Returns the number of policy rules in the rule base.
getLastRebuild()	Returns the timestamp the policy rule base was last rebuilt. The rule base is naturally rebuilt whenever the system determines that a policy or a related policy object has been created, modified, or deleted.
rebuildRuleBase()	Rebuilds the current server's policy rule base.

com.twowire.dmc.service.impl/lockService

The `lockService` MBean exposes management information for the lock service that controls which server owns which global locks. The MBean has the following operations:

Operation	Description
getOwnedLockNames()	Returns the names of the locks owned by this server.
relinquishLocks()	Forces the server to relinquish its owned locks, at which point they may be obtained by another server in the cluster.

com.twowire.dmc.service.impl/scheduler

The `scheduler` manages the execution of scheduled jobs, specifically firmware upgrade scheduled workflows. The MBean exposes the following management operations:

Operation	Description
getState()	Returns the state of the scheduler, such as whether it is started or stopped.

Index