IBM® Tivoli® Netcool/OMNIbus Probe for Huawei U2000 3GPP (CORBA)
2.0

Reference Guide
July 20, 2017

SC27-6583-03

Notice

Before using this information and the product it supports, read the information in Appendix A, “Notices and Trademarks,” on page 39.

Edition notice

This edition (SC27-6583-03) applies to version 2.0 of IBM Tivoli Netcool/OMNIbus Probe for Huawei U2000 3GPP (CORBA) and to all subsequent releases and modifications until otherwise indicated in new editions.

This edition replaces SC27-6583-02.

Contents

About this guide

The following sections contain important information about using this guide.

Document control page

Use this information to track changes between versions of this guide.

The IBM Tivoli Netcool/OMNIbus Probe for Huawei U2000 3GPP (CORBA) documentation is provided in softcopy format only. To obtain the most recent version, visit the IBM® Tivoli® Knowledge Center:

https://www.ibm.com/support/knowledgecenter/SSSHTQ/omnibus/probes/common/Probes.html

Table 1. Document modification history

Document version	Publication date	Comments
SC27-6583-00	August 6, 2015	First IBM publication.
SC27-6583-01	February 19, 2016	Updated to reflect support extended to the following platform: Huawei iManager U2000 V200 R015
SC27-6583-02	March 14, 2017	Updated for version 2.0 of the Probe for Huawei U2000 3GPP (CORBA). "Summary" on page 1 updated. "Customizing batch sizes for resynchronization" on page 6 updated. Description for the **ResyncBatchSize** property added to "Properties and command line options" on page 15 New issue regarding displaying the probe's release ID added to "Known issues" on page 31. **Enhancements:** Version 2.0 of the probe addresses the following enhancement requests: • **93383/95631**: Extended support to Huawei iManager U2000 V200 R016 **Fixes:** Version 2.0 of the probe includes fixes for the following APARs: • **IV79240**: Initial resynchronization not working. • **IV77999**: Probe now parses the `RemainderOfBody` element into specific attribute names. • **IV80198**: Probe not displaying the correct IRP classname.
SC27-6583-03	July 20, 2017	Details about encrypting passwords corrected in the following topics: • "SSL-based connectivity" on page 3 • "Encrypting passwords" on page 15 • "Properties and command line options" on page 15

Conventions used in this guide

All probe guides use standard conventions for operating system-dependent environment variables and directory paths.

Operating system-dependent variables and paths

All probe guides use standard conventions for specifying environment variables and describing directory paths, depending on what operating systems the probe is supported on.

For probes supported on UNIX and Linux operating systems, probe guides use the standard UNIX conventions such as **$***variable* for environment variables and forward slashes (**/**) in directory paths. For example:

`$OMNIHOME/probes`

For probes supported only on Windows operating systems, probe guides use the standard Windows conventions such as **%***variable***%** for environment variables and backward slashes (****) in directory paths. For example:

`%OMNIHOME%\probes`

For probes supported on UNIX, Linux, and Windows operating systems, probe guides use the standard UNIX conventions for specifying environment variables and describing directory paths. When using the Windows command line with these probes, replace the UNIX conventions used in the guide with Windows conventions. If you are using the bash shell on a Windows system, you can use the UNIX conventions.

Note : The names of environment variables are not always the same in Windows and UNIX environments. For example, %TEMP% in Windows environments is equivalent to $TMPDIR in UNIX and Linux environments. Where such variables are described in the guide, both the UNIX and Windows conventions will be used.

Operating system-specific directory names

Where Tivoli Netcool/OMNIbus files are identified as located within an *arch* directory under NCHOME or OMNIHOME, *arch* is a variable that represents your operating system directory. For example:

`$OMNIHOME/probes/`*`arch`*

The following table lists the directory names used for each operating system.

Note : This probe may not support all of the operating systems specified in the table.

Table 2. Directory names for the arch variable

Operating system	**Directory name represented by *arch***
AIX® systems	`aix5`
HP-UX PA-RISC-based systems	`hpux11`
Red Hat Linux® and SUSE systems	`linux2x86`
Linux for System z®	`linux2s390`
Solaris systems	`solaris2`

OMNIHOME location

Probes and older versions of Tivoli Netcool/OMNIbus use the OMNIHOME environment variable in many configuration files. Set the value of OMNIHOME as follows:

- On UNIX and Linux, set $OMNIHOME to $NCHOME/omnibus.
- On Windows, set %OMNIHOME% to %NCHOME%\omnibus.

Chapter 1. Probe for Huawei U2000 3GPP (CORBA)

Huawei U2000 3GPP provides operators with a centralized, mobile network operation and maintenance (O&M) platform.

The device is used for the management of mobile network elements such as WRAN, GBSS, SingleRAN, core network, LTE/EPC, NGN, STP, IMS, and other transmission equipment at both the network element (NE) layer and the network layer.

The IBM Tivoli Netcool/OMNIbus Probe for Huawei U2000 3GPP (CORBA) (formerly known as Huawei M2000 (CORBA)) collects events from the Huawei U2000 3GPP management system using a 3rd Generation Partnership Project (3GPP) Common Object Request Broker Architecture (CORBA) interface.

This guide contains the following sections:

Summary

Each probe works in a different way to acquire event data from its source, and therefore has specific features, default values, and changeable properties. Use this summary information to learn about this probe.

The following table summarizes the probe.

Table 3. Summary

Probe target	Huawei iManager M2000 V200 R013 Huawei iManager U2000 V200 R014 Huawei iManager U2000 V200 R015 Huawei iManager U2000 V200 R016 **Note :** The EMS versions R015 and R015 have SSL enabled by default. Before running the probe, you should either disable SSL within the EMS, or enable SSL connectivity in the probe. For details see "SSL-based connectivity" on page 3.
Probe executable name	`nco_p_huawei_u2000_3gpp_corba`

Table 3. Summary (continued)	
Probe installation package	`omnibus-`*`arch`*`-probe-nco-p-huawei-u2000-3gpp-corba-`*`version`*
Package version	2.0
Probe supported on	For details of supported operating systems, see the following Release Notice on the IBM Software Support website: https://www-304.ibm.com/support/docview.wss?uid=swg21579706
Properties file	`$OMNIHOME/probes/`*`arch`*`/huawei_u2000_3gpp_corba.props`
Rules file	`$OMNIHOME/probes/`*`arch`*`/huawei_u2000_3gpp_corba.rules`
Requirements	For details of any additional software that this probe requires, refer to the `description.txt` file that is supplied in its download package.
Connection method	CORBA
Remote connectivity	The probe can connect to a remote device using the CORBA interface.
Multicultural support	Available
Peer-to-peer failover functionality	Available
IP environment	IPv4 and IPv6
Federal Information Processing Standards (FIPS)	IBM Tivoli Netcool/OMNIbus uses the FIPS 140-2 approved cryptographic provider: IBM Crypto for C (ICC) certificate 384 for cryptography. This certificate is listed on the NIST website at http://csrc.nist.gov/groups/STM/cmvp/documents/140-1/1401val2004.htm. For details about configuring Netcool/OMNIbus for FIPS 140-2 mode, see the *IBM Tivoli Netcool/OMNIbus Installation and Deployment Guide*.

Installing probes

All probes are installed in a similar way. The process involves downloading the appropriate installation package for your operating system, installing the appropriate files for the version of Netcool/OMNIbus that you are running, and configuring the probe to suit your environment.

The installation process consists of the following steps:

1. Downloading the installation package for the probe from the Passport Advantage Online website.

   Each probe has a single installation package for each operating system supported. For details about how to locate and download the installation package for your operating system, visit the following page on the IBM Tivoli Knowledge Center:

   http://www-01.ibm.com/support/knowledgecenter/SSSHTQ/omnibus/probes/all_probes/wip/reference/install_download_intro.html

2. Installing the probe using the installation package.

   The installation package contains the appropriate files for all supported versions of Netcool/OMNIbus. For details about how to install the probe to run with your version of Netcool/OMNIbus, visit the following page on the IBM Tivoli Knowledge Center:

   http://www-01.ibm.com/support/knowledgecenter/SSSHTQ/omnibus/probes/all_probes/wip/reference/install_install_intro.html

3. Configuring the probe.

   This guide contains details of the essential configuration required to run this probe. It combines topics that are common to all probes and topics that are peculiar to this probe. For details about additional configuration that is common to all probes, see the *IBM Tivoli Netcool/OMNIbus Probe and Gateway Guide*.

Firewall considerations

When using CORBA probes in conjunction with a firewall, the firewall must be configured so that the probe can connect to the target system.

Most CORBA probes can act as both a server (listening for connections from the target system) and a client (connecting to the port on the target system to which the system writes events). If you are using the probe in conjunction with a firewall, you must add the appropriate firewall rules to enable this dual behavior.

There are three possible firewall protection scenarios, for which you must determine port numbers before adding firewall rules:

1. If the host on which the probe is running is behind a firewall, you must determine what remote host and port number the probe will connect to.
2. If the host on which the target system is running is behind a firewall, you must determine the incoming port on which the probe will listen and to which the target system will connect.
3. If each host is secured with its own firewall, you must determine the following four ports:
   a. The outgoing port (or port range) for the probe.
   b. The hostname and port of the target system.
   c. The outgoing port on which the target system sends events if the probe is running as a client.
   d. The incoming port on which the probe listens for incoming events.

Note : Most, but not all, CORBA probes listen on the port specified by the **ORBLocalPort** property. The default value for this property is 0, which means that an available port is selected at random. If the probe is behind a firewall, the value of the **ORBLocalPort** property must be specified as a fixed port number.

CORBA probes that use EventManager or NotificationManager objects may use different hosts and ports from those that use NamingService and EntryPoint objects. If the probe is configured to get object references from a NamingService or EntryPoint object, you must obtain the host and port information from the system administrator of the target system. When you have this information, you can add the appropriate firewall rules.

SSL-based connectivity

The Probe for Huawei U2000 3GPP (CORBA) supports Secure Sockets Layer (SSL) connections between the probe and the EMS server. SSL connections provide additional security when the probe retrieves alarms from the EMS.

To enable SSL connections, obtain the required SSL certificates and the Trusted Authority certificate from the EMS vendor, Huawei Technologies. Add the certificates to a local Java™ keystore so that they can be referenced by the **KeyStore** property.

Prerequisites

To create the keystore, ensure you have the following software installed:

- The OpenSSL toolkit.

  This is available from http://www.openssl.org/.
- The IBM KeyMan utility.

  This is available from http://www.alphaworks.ibm.com/tech/keyman/download.

You must also obtain the client and server certificates, `client_ca.cer` and `server_ca.cer`, and the server key pair, `server_key.pem`, from Huawei Technologies.

Note : The certificate and key pair files used here are the default files used by the Huawei U2000 EMS. If you replace these files, you must create a keystore containing the new files.

Creating the SSL keystore and truststore

To create a Java keystore, follow these steps:

1. Convert the server certificate to PKCS12 format using the following OpenSSL toolkit command:

   ```
   openssl pkcs12 -export -inkey server_key.pem -in server_ca.cer -out
   server_ca.pkcs12
   ```
2. Create the keystore using the KeyMan utility:
   a. Start the KeyMan utility.
   b. Click **Create New** and select the **Keystore token** option.
   c. Click **File** > **Import** and choose the `server_ca.pkcs12` file that you created in step 1.

      This imports the `keyEntry` into the keystore.
   d. Click **File** > **Import** and choose the `server_ca.cer` certificate.

      This imports the server certificate into the keystore.
   e. Click **File** > **Import** and choose the `client_ca.cer` certificate.

      This imports the client certificate into the keystore.
   f. Click **File** > **Save** and enter a password and name for the keystore, for example *`trusted_keystore`*`.jks`.

Enabling SSL connections

To enable SSL-based connections between the probe and the EMS server, follow these steps:

1. Configure the probe connection methods to use the Naming service by configuring the Naming service host and port or Naming service IOR file.
2. Set the **EnableSSL** property to `true`.

   When the **EnableSSL** property is set to `true`, the following properties are enabled:

   - **KeyStore**
   - **KeyStorePassword**
   - **SecurityProtocol**
3. Use the **KeyStore** property to specify the location of the keystore file *`trusted_keystore`*`.jks`.
4. Use the **KeyStorePassword** property to specify a password for the keystore.
5. Encrypt the keystore file password (if required) using the encryption utility supplied with Netcool/OMNIbus. See "Encrypting passwords" on page 15.

Data acquisition

Each probe uses a different method to acquire data. Which method the probe uses depends on the target system from which it receives data.

The probe gathers events from the Huawei U2000 3GPP system using a 3GPP CORBA interface.

Data acquisition is described in the following topics:

Connecting to the CORBA interface

The probe connects to the CORBA interface using Interoperable Object Reference (IOR) files.

The probe gets the object reference of the `AlarmIRPOperation` and `NotificationIRPOperation` objects from the IOR files specified by the **`AlarmIRPName`** and **`NotificationIRPName`** properties. The `AlarmIRPOperation` and `NotificationIRPOperation` objects form a part of the Integration Reference Point (IRP) Agent.

To get the IOR files, the probe retrieves the Entry Point IRP IOR file specified by the **`EntryPointIORFile`** property.

To define the alarm and notification IRP IOR files, the probe retrieves the information specified in the **`AlarmIRPIORFile`** and **`NotificationIRPIORFile`** properties.

To use the Naming service, provide the host and port number by specifying the **`NamingServiceHost`** and **`NamingServicePort`** properties.

The Naming Service IOR file is specified by the **`NamingServiceIORFile`** property.

If Entry Point IRP authentication is enabled on the IRP Agent, you must use the **`Username`** and **`Password`** properties to specify the user name and password required to log in to the Huawei U2000 3GPP system.

Note : If an encrypted password has been configured in the Huawei U2000 3GPP EMS for the RAN Sharing feature, you will have to obtain the encrypted password string from the EMS vendor, Huawei Technologies.

Retrieving objects

When the **`InitialResync`** property is set to true, the probe initially receives a list of all active alarms from the `AlarmIRP` object.

The probe then connects to the `NotificationIRP` object and uses the CORBA notification push model to receive new alarms from the server as they are generated. When the **`InitialResync`** property is set to `false`, the probe receives only new alarms.

Customizing batch sizes for resynchronization

You can specify the maximum number of alarms contained in each batch that the probe receives during resynchronization.

To customize the batch size for resynchronization, use the **ResyncBatchSize** property to specify the maximum number of alarms contained in each batch that the probe receives during a resynchronization operation.

Note : The minimum value for this property is 1.

IRP Agent status

The probe checks the status of the IRP Agent every 60 seconds.

You can use the **HeartbeatInterval** property to change the frequency with which the probe checks the subscription status of the Notification IRP Agent.

Backoff strategy

If the **RetryCount** property is set to `true`, and the probe fails to establish a connection or loses an existing connection to the device, it reverts to a backoff strategy. The probe uses backoff strategy as mentioned (1s, 2s, 4s etc) if **RetryInterval** is set to `0`. If **RetryInterval** > 0, the probe will retry at a fixed interval period. Once the connection is made to the CORBA interface, the probe tries to log in to the device. If the probe fails to log in, it shuts down and tries to connect again. The backoff strategy remains in place until a successful login occurs.

Inactivity

The probe can disconnect from the target system and shut down if there is no event activity for a predefined amount of time.

You can use the **Inactivity** property to specify how long, in seconds, the probe waits before disconnecting from the target system and shutting down. If the probe receives no events during that time, it disconnects from the target system and shuts down. To ensure that the probe never disconnects from the target system, set the value of the property to `0`, which is the default value.

Filtering notifications and alarms

You can use filtering properties to specify how the probe filters events to receive only preferred notifications and alarms.

The **NotificationFilter** and **ResyncFilter** properties allow you to specify what notifications and alarms are sent to the probe. When you use these properties, specify the actual token names. For example, the token h represents the element `NV_PERCEIVED_SEVERITY`. To specify that the probe is only sent notifications with a perceived severity of 3, set the **ResyncFilter** property to `$h == 3`.

You can specify more complex filters using `or` and `and` operators. For example, to specify that the probe receives notifications with a perceived severity of 3 or 4, set the **ResyncFilter** property to the following value:

```
$h == 3 or $h == 4
```

You can use the tokens listed below as notification filter properties.

To specify that the probe is only sent with communications alarms, set the **NotificationFilter** property to `$event_name == 'x1'`.

Commonly supported tokens

`$type_name`: This token represents the type of notification and allows the following possible values:

`'x1'`: New alarm

'x3': Alarm with its state changed to either acknowledged or unacknowledged

'x5': Cleared alarm

$event_name: This token represents the type of alarm and allows the following possible values:

'x1': Communications alarm

'x2': Processing error alarm

'x3': Environmental alarm

'x4': Quality of service alarm

'x5': Equipment alarm

'x6': Integrity violation

'x7': Operational violation

'x8': Physical violation

'x9': Security service or mechanism violation

'x10': Time domain violation

Token supported in **NotificationFilter** only:

$e: This token represents the managed object instance.

Tokens supported in **ResyncFilter** only

$b: This token represents event time in units of 100 nano seconds since October 15, 1582.

$h: This token represents perceived severity of alarm and allows the following possible values:

2: Critical

3: Major

4: Minor

5: Warning

Lookup table

The probe is supplied with a lookup table that contains details of the various types of alarms that the Huawei U2000 3GPP generates.

At installation, the Corba_3gpp_V630.lookup file supplied with the probe installation package is installed to the following location:

$OMNIHOME/probes/includes/

The following line in the rules file references the lookup file:

```
include "../includes/Corba_3gpp_V630.lookup"
```

Note : The include command assumes that the probe is run from the standard $OMNIHOME/probes/ directory. If you are running the probe from a different directory, replace ".." with the absolute directory path to the lookup file. Do not use the $OMNIHOME environment variable in this directory path.

Persistent subscription

When the **PersistentUnsubscribe** property is set to true, the probe will unsubscribe an old session before subscribing to a new session.

The probe stores subscription information in a temporary file. If the probe is disconnected or shut down ungracefully, the probe will check against this temp file and detect the old subscription during the next probe start up. The probe will then attempt to unsubscribe the old session and remove the old subscription file before proceeding to subscribe to a new session.

If multiple probe instances exists, refer to the Name property to differentiate the instances.

When the property is set to `false`, the probe will only unsubscribe an old session when shut down. Subscription information will not be stored.

Peer-to-peer failover functionality

The Probe for Huawei U2000 3GPP (CORBA) supports failover configurations where two probes run simultaneously. One probe acts as the `master` probe, sending events to the ObjectServer; the other acts as the `slave` probe on standby. If the master probe fails, the slave probe activates.

While the slave probe receives heartbeats from the master probe, it will not forward events to the ObjectServer. If the master shuts down, the slave probe will stop receiving heartbeats from the master and any events it receives thereafter will be forwarded to the ObjectServer on behalf of the master probe. When the master is running again, the slave will continue to receive events, but will no longer send them to the ObjectServer.

Example property file settings for peer-to-peer failover

You set the peer-to-peer failover mode in the properties files of the master and slave probes. The settings differ for a master probe and slave probe.

Note : In the examples, make sure to use the full path for the property value. In other words replace $OMNIHOME with the full path. For example: `/opt/IBM/tivoli/netcool`.

The following example shows the peer-to-peer settings from the properties file of a master probe:

```
Server      :    "NCOMS"
RulesFile   :    "master_rules_file"
MessageLog  :    "master_log_file"
PeerHost    :    "slave_hostname"
PeerPort    :    5555 # [communication port between master and slave probe]
Mode        :    "master"
PidFile     : "master_pid_file"
```

The following example shows the peer-to-peer settings from the properties file of the corresponding slave probe:

```
Server      :    "NCOMS"
RulesFile   :    "slave_rules_file"
MessageLog  :    "slave_log_file"
PeerHost    :    "master_hostname"
PeerPort    :    5555 # [communication port between master and slave probe]
Mode        :    "slave"
PidFile     : "slave_pid_file"
```

Huawei U2000 3GPP server redundancy

Two Huawei U2000 3GPP EMS servers can run in a redundancy pair (that is, one runs as the primary server and the other as a backup server). This affects the way that you configure the probe.

If you are using two Huawei U2000 3GPP servers in a redundancy pair, you can configure one probe instance that will fail-over to the primary and secondary EMS. These settings can be found in the properties file.

Configuring probe for server redundancy

If using AlarmIRP and NotificationIRP IOR file, configure the following :

- **`EnableFailover`** set to `true`
- **`RetryCount`** set > 0
- **`AlarmIRPIORFile`** and **`SecondaryAlarmIRPIORFile`**

- **NotificationIRPIORFile** and **SecondaryNotificationIRPIORFile**

If using Naming Service host and port, configure the following:

- **EnableFailover** set to `true`
- **RetryCount** set > 0
- **NamingServiceHost** and **SecondaryNamingServiceHost**
- **NamingServicePort** and **SecondaryNamingServicePort**
- **AlarmIRPName** and **SecondaryAlarmIRPName**
- **NotificationIRPName** and **SecondaryNotificationIRPName**

If using Naming Service IOR file, configure the following:

- **EnableFailover** set to `true`
- **RetryCount** set > 0
- **NamingServiceIORFile** and **SecondaryNamingServiceIORFile**
- **AlarmIRPName** and **SecondaryAlarmIRPName**
- **NotificationIRPName** and **SecondaryNotificationIRPName**

If using EntryPoint IOR File, configure the following :

- **EnableFailover** set to `true`
- **RetryCount** set > 0
- **EntryPointIORFile** and **SecondaryEntryPointIORFile**
- **AlarmIRPName** and **SecondaryAlarmIRPName**
- **NotificationIRPName** and **SecondaryNotificationIRPName**

If using Naming Service host and port to connect through SSL, configure the following:

- **EnableFailover** set to `true`
- **RetryCount** set > 0
- **NamingServiceHost** and **SecondaryNamingServiceHost**
- **NamingServicePort** and **SecondaryNamingServicePort**
- **AlarmIRPName** and **SecondaryAlarmIRPName**
- **NotificationIRPName** and **SecondaryNotificationIRPName**
- **EnableSSL** set to `true`
- **KeyStore**
- **KeyStorePassword**

Note : The primary and secondary EMS security certificates may need to be imported into the same JKS and defined in the KeyStore.

Command Line Interface

The probe is supplied with a Command Line Interface (CLI). This interface enables you to execute commands to acknowledge alarms or request a full resynchronization of the 3GPP interface.

To use the CLI, you must use the **CommandPort** property to specify a port through which commands will be sent. The default port is 6970. When you want to issue commands, use Telnet to connect through this port. You can use the **CommandPortLimit** property to limit the number of Telnet connections that the probe can make at one time.

The following table describes the commands that you can use with the CLI.

Table 4. CLI commands	
Command	**Description**
exit **quit**	Use these commands to close the connection.
help	Use this command to display online help about the CLI.
resynch_all	Use this command to perform a full resynchronization with the CORBA interface.
resynch_filter *filter*	Use this command to perform partial resynchronization with the CORBA interface. The *filter* parameter uses the same filter format as the **ResyncFilter** property. For example: `> resynch_filter $h==3`
userid_acknowledge_alarm *alarm_id user_id*	Use this command to acknowledge an alarm in the CORBA interface by specifying the alarm ID (NV_ALARM_ID) and the user ID of the user acknowledging the alarm. If you specify a value for the **AckUserId** property, the *user_id* parameter is not required.
userid_unacknowledge_alarm *alarm_id user_id*	Use this command to unacknowledge an alarm in the CORBA interface by specifying the alarm ID (NV_ALARM_ID) and the user ID of the user acknowledging the alarm. If you specify a value for the **AckUserId** property, the *user_id* parameter is not required.
userid_clear_alarm *alarm_id user_id*	Use this command to clear an alarm by specifying the alarm ID (NV_ALARM_ID) and the user ID of the user acknowledging the alarm. If you specify a value for the **ClearUserId** property, the *user_id* parameter is not required.
userid_comment_alarm *alarm_id user_id comment_text*	Use this command to add comments to an alarm by specifying the alarm ID (NV_ALARM_ID), the user ID of the user making the comment, and the text of the comment. If you specify a value for the **CommentUserId** property, the *user_id* parameter is not required. **Note :** There is a known issue with comments longer than one word being logged as the user ID in the log file. For details, see "Known issues" on page 31.
version	Use this command to display the version of the probe.

CLI scripts

Because the CLI uses Telnet connections, you can connect to the probe from anywhere by creating a desktop tool to open a Telnet connection, send a command, and then close the connection. This means that simple scripts can be set up to allow users to acknowledge selected events from the IBM Tivoli Netcool/OMNIbus event list.

HTTP/HTTPS command interface

IBM Tivoli Netcool/OMNIbus Version 7.4.0 (and later) includes a facility for managing the probe over an HTTP/HTTPS connection. This facility uses the **nco_http** utility supplied with Tivoli Netcool/OMNIbus.

The HTTP/HTTPS command interface replaces the Telnet-based command line interface used in previous versions of IBM Tivoli Netcool/OMNIbus.

The following sections show:

- How to configure the command interface.
- The format of the **nco_http** command line.
- The format of the individual probe commands.
- The messages that appear in the log files.
- How to store frequently-used commands in a properties file.

For more information on the HTTP/HTTPS command interface and the utilities it uses, see the chapter on remotely administering probes in the *IBM Tivoli Netcool/OMNIbus Probe and Gateway Guide*.

Configuring the command interface

To configure the HTTP/HTTPS command interface, set the following properties in the probe's property file:

NHttpd.EnableHTTP: Set this property to `True`.
NHttpd.ListeningPort: Set this property to the number of the port that the probe uses to listen for HTTP commands.

Optionally, set a value for the following property as required:

NHttpd.ExpireTimeout: Set this property to the maximum time (in seconds) that the HTTP connection remains idle before it is disconnected.

The *IBM Tivoli Netcool/OMNIbus Probe and Gateway Guide* contains a full description of these and all properties for the HTTP/HTTPS command interface.

Format of the nco_http command line

The format of the **nco_http** command line to send a command to the probe is:

```
$OMNIHOME/bin/nco_http -uri probeuri:probeport/probes/huawei_u2000_3gpp_corba -
datatype application/json -method post -data '{"command":"command-
name","params":[command-parameters]}'
```

Where:

- *probeuri* is the URI of the probe.
- *probeport* is the port that the probe uses to listen for HTTP/HTTPS commands. Specify the same value as that set for the **NHttp.ListeningPort**.
- *command-name* is the name of the command to send to the probe. The following command names are available:

  help
  resynch_all
  resynch_filter

userid_acknowledge_alarm
userid_unacknowledge_alarm
userid_clear_alarm
userid_comment_alarm
version

- *command-parameters* is a list of zero or more command parameters. For commands that have no parameters, this component is empty. The command descriptions in the following section define the parameters that each takes.

Probe commands

The following sections define the structure of the JavaScript Object Notation (JSON)-formatted commands that you can send to the probe. There is an example of each command.

All the examples use a probe URI of `http://localhost` and a HTTP listening port of `8080`.

help

Use the **help** command to receive help information about the HTTP/HTTPS command interface.

The format of the `-data` option for the **help** command is:

```
-data '{"command":"help","params":[]}'
```

The following command returns help information:

```
$OMNIHOME/bin/nco_http -uri http://localhost:56100/probes/
huawei_u2000_3gpp_corba -datatype application/JSON -method POST -data
'{"command":"help","params":[]}'
```

resynch_all

Use the **resynch_all** command to perform a full resynchronization with the CORBA interface.

The format of the `-data` option for the **resynch_all** command is:

```
-data '{"command":"resynch_all","params":[]}'
```

The following command performs a resynchronization:

```
$OMNIHOME/bin/nco_http -uri http://localhost:56100/probes/
huawei_u2000_3gpp_corba -datatype application/JSON -method POST -data
'{"command":"resynch_all","params":[]}'
```

resynch_filter

Use the **resynch_filter** command to perform a partial resynchronization with the CORBA interface using a custom filter.

The format of the `-data` option for the **resynch_filter** command is:

```
-data '{"command":"resynch_filter","params":[{"filter":"($h == 3) OR ($h ==
4)"}]}'
```

Note : The HTTP command interface does not suffer the limitation of containing spaces in the filter input.

The following command performs a resynchronization:

```
$OMNIHOME/bin/nco_http -uri http://localhost:56100/probes/
huawei_u2000_3gpp_corba -datatype application/JSON -method POST -data
'{"command":"resynch_filter", "params":[{"filter":"($h == 3) OR ($h == 4)"}]}'
```

userid_acknowledge_alarm

Use the **userid_acknowledge_alarm** command to acknowledge an alarm in the CORBA interface by specifying the Alarm ID (NV_ALARM_ID) and the User ID of the user acknowledging the alarm. If you specify a value for the **AckUserId** property, the *user_id* parameter is not required.

The format of the -data option for the **userid_acknowledge_alarm** command is:

```
-data '{"command":"userid_acknowledge_alarm", "params":
[{"alarm_id":"exalarmid","user_id":"exuserid"}]}'
```

Where:

- *exalarmid* is the alarm ID to be acknowledged.
- *exuserid* is the user ID that acknowledges the alarm.

The following command acknowledges an alarm:

```
$OMNIHOME/bin/nco_http -uri http://localhost:56100/probes/
huawei_u2000_3gpp_corba -datatype application/JSON -method POST -data
'{"command":"userid_acknowledge_alarm", "params":
[{"alarm_id":"abc123","user_id":"netcool"}]}'
```

userid_unacknowledge_alarm

Use the **userid_unacknowledge_alarm** command to unacknowledge an alarm in the CORBA interface by specifying the alarm ID (NV_ALARM_ID) and the user ID of the user acknowledging the alarm. If you specify a value for the **AckUserId** property, the *user_id* parameter is not required

The format of the -data option for the **userid_unacknowledge_alarm** command is:

```
-data '{"command":"userid_unacknowledge_alarm", "params":
[{"alarm_id":"exalarmid","user_id":"exuserid"}]}'
```

Where:

- *exalarmid* is the alarm ID to be acknowledged.
- *exuserid* is the user ID that acknowledges the alarm.

The following command unacknowledges an alarm:

```
$OMNIHOME/bin/nco_http -uri http://localhost:56100/probes/
huawei_u2000_3gpp_corba -datatype application/JSON -method POST -data
'{"command":"userid_unacknowledge_alarm", "params":
[{"alarm_id":"abc123","user_id":"netcool"}]}'
```

userid_clear_alarm

Use the **userid_clear_alarm** command to clear an alarm by specifying the alarm ID (NV_ALARM_ID) and the user ID of the user acknowledging the alarm. If you specify a value for the **ClearUserId** property, the *user_id* parameter is not required.

The format of the -data option for the **userid_clear_alarm** command is:

```
-data '{"command":"userid_clear_alarm", "params":
[{"alarm_id":"exalarmid","user_id":"exuserid"}]}'
```

Where:

- *exalarmid* is the alarm ID to be acknowledged.
- *exuserid* is the user ID that acknowledges the alarm.

The following command clears an alarm:

```
$OMNIHOME/bin/nco_http -uri http://localhost:56100/probes/
huawei_u2000_3gpp_corba -datatype application/JSON -method POST -data
```

```
'{"command":"userid_clear_alarm", "params":
[{"alarm_id":"abc123","user_id":"netcool"}]}'
```

userid_comment_alarm

Use the **userid_comment_alarm** command to add comments to an alarm by specifying the alarm ID (NV_ALARM_ID), the user ID of the user making the comment, and the comment text. If you specify a value for the *CommentUserId* property, the *user_id* parameter is not required.

The format of the -data option for the **userid_comment_alarm** command is:

```
-data '{"command":"userid_comment_alarm", "params":[{"alarm_id":"exalarmid",
"user_id":"exuserid", "comment_text":"extext"}]}'
```

Where:

- *exalarmid* is the alarm ID to be acknowledged.
- *exuserid* is the user ID that acknowledges the alarm.
- *extext* is the text commenting on the alarm.

The following command adds comments to an alarm:

```
$OMNIHOME/bin/nco_http -uri http://localhost:56100/probes/
huawei_u2000_3gpp_corba -datatype application/JSON -method POST -data
'{"command":"userid_comment_alarm", "params":[{"alarm_id":"abc123",
"user_id":"netcool","comment_text":"test"}]}'
```

version

Use the **version** command to print the version of the probe.

The format of the -data option for the **version** command is:

```
-data '{"command":"version","params":[]}'
```

The following command returns version information:

```
$OMNIHOME/bin/nco_http -uri http://localhost:56100/probes/
huawei_u2000_3gpp_corba -datatype application/JSON -method POST -data
'{"command":"version","params":[]}'
```

Messages in the log file

The nco_http utility can make extensive entries in the probe's log file indicating the progress of each operation. These messages can help isolate problems with a request, such as a syntax problem in a command.

To obtain the detailed log information, set the probe's **MessageLevel** property to debug. This enables the logging of the additional information that tracks the progress of a command's execution. For example, the following shows the progress of a **resynch_all** command:

```
Information: I-UNK-104-002: {"response":["resynch_all successfully
completed. Resynchronization OK"],"status":"200"}
```

Storing commands in the nco_http properties file

You can use the **nco_http** utility's properties file ($OMNIHOME/etc/nco_http.props) to hold frequently used command characteristics.

If you have a particular command that you send to the probe regularly, you can store characteristics of that command in the **nco_http** properties file. Once you have done that, the format of the **nco_http** command line is simplified.

You can use one or more of the following **nco_http** properties to hold default values for the equivalent options on the **nco_http** command line:

Data
DataType
Method
URI

Specify the value of each property in the same way as you would on the command line. Once you have these values in place you do not need to specify the corresponding command line switch unless you want to override the value of the property.

The following is an example of the use of the properties file and the simplification of the **nco_http** command that results. In this example, the **nco_http** properties file contains the following values (note that line breaks appear for presentational purposes only; when editing the properties use one line for each property value):

```
Data : [example required]
DataType : 'application/JSON'
Method : 'POST'
```

Encrypting passwords

You can encrypt the passwords specified by the **Password** property and the **KeyStorePassword** property using the `nco_aes_crypt` utility supplied with Netcool/OMNIbus.

If you encrypt the passwords using `nco_aes_crypt`, you must also specify values for the **ConfigCryptoAlg** and **ConfigKeyFile** properties. For example:

```
ConfigCryptoAlg: 'AES_FIPS'
ConfigKeyFile: '$NCHOME/omnibus/etc/keyfile'
```

Properties and command line options

You use properties to specify how the probe interacts with the device. You can override the default values by using the properties file or the command line options.

The following table describes the properties and command line options specific to this probe. For more information about generic Netcool/OMNIbus properties and command line options, see the *IBM Tivoli Netcool/OMNIbus Probe and Gateway Guide*.

Table 5. Properties and command line options

Property name	Command line option	Description
AckUserId *string*	`-ackuserid` *string*	Use this property to specify the user ID to use when acknowledging and unacknowledging alarms using the CLI. If you do not use the *user_id* parameter when acknowledging or unacknowledging an alarm using the CLI, the probe will add the user ID specified by this property to the related log entry. The default is " ".
AlarmIRPIORFile *string*	`-alarmirpiorfile` *string*	Use this property to specify the Alarm IRP IOR file. The default is " ".

Table 5. Properties and command line options (continued)		
Property name	**Command line option**	**Description**
AlarmIRPName *string*	`-alarmirpname` *string*	Use this property to specify the name that the probe uses to resolve the AlarmIRP point. The default is `AlarmIRP`.
ClearUserId *string*	`-clearuserid` *string*	Use this property to specify the user ID to use when clearing alarms using the CLI. If you do not use the *user_id* parameter when clearing an alarm using the CLI, the probe will add the user ID specified by this property to the related log entry. The default is `" "`.
CommentUserId *string*	`-commentuserid` *string*	Use this property to specify the user ID to use when adding comments to alarms using the CLI. If you do not use the *user_id* parameter when commenting on an alarm using the CLI, the probe will add the user ID specified by this property to the related log entry. The default is `" "`. **Note :** There is a known issue with comments longer than one word being logged as the user ID in the probe log file, regardless of the value of this property. For details, see "Known issues" on page 31.
EnableFailover *string*	`-enablefailover` *string*	Use this to enable rotate endpoint for EMS server failover. The default is `false`.
EnableSSL *string*	`-disablessl` (This is equivalent to **EnableSSL** with a value of `false`.) `-enablessl` (This is equivalent to **EnableSSL** with a value of `true`.)	Use this property to specify whether SSL connectivity between the probe and the server is enabled or disabled. This property takes the following values: `false`: SSL connectivity between the probe and the server is disabled. `true`: SSL connectivity between the probe and the server is enabled. The default is `false`.

Table 5. Properties and command line options (continued)		
Property name	**Command line option**	**Description**
EncodingStandard *string*	`-encodingstandard` *string*	Use this property to specify the encoding required to encode and decode characters in events. Specify a value of UTF8 for this property to enable the probe to receive events in languages that use multibyte characters, such as Chinese or Japanese. The default is `ISO-8859-1` (also known as Latin-1).
EntryPointIORFile *string*	`-entrypointiorfile` *string*	Use this property to specify the location of the Entry Point IRP IOR file. The default is `" "`.
IDLAttrMapFile *string*	`-idlattrmapfile` *string*	Use this property to specify the CORBA IDL attributes mapping in an .xml format file. The default is `$OMNIHOME/probes/includes/huawei_u2000_3gpp_RuleElementMap.xml`. **Note :** Replace the default value with a full path directory file name . Environment variables such as $OMNIHOME or %OMNIHOME% is not recognized by the probe for this property.
IrpVersion *string*	`-Irpvesion` *string*	Use this property to specify the version identifier required to get the IRP outline. The default is `V1.2.0`.
KeyStore *string*	`-keystore` *string*	Use this property to specify the location of the keystore file that contains the client certificate for SSL and trusted authority certificate. The default is `""`.
KeyStorePassword *string*	`-keystorepassword` *string*	Use this property to specify the password required to access the certificate defined in the **Keystore** property. The default is `""`. **Note :** You can encrypt the keystore file password using the encryption utility supplied with Netcool/OMNIbus. See “Encrypting passwords” on page 15.

Table 5. Properties and command line options (continued)		
Property name	**Command line option**	**Description**
ManagerIdentifier *string*	`-manageridentifier` *string*	Use this property to specify the name used to get all the IRP references. The default is `" "`.
NamingServiceHost *string*	`-namingservicehost` *string*	Use this property to specify the Naming Service host. The default is `localhost`.
NamingServiceIORFile *string*	`-namingserviceiorfile` *string*	Use this property to specify the Naming Service IOR file. The default is `""`.
NamingServicePort *integer*	`-namingserviceport` *integer*	Use this property to specify the Naming Service port number. The default is `0`.
Notification Categories *string*	`-notificationcategories` *string*	Use this property to specify the notification categories to which the probe subscribes. To specify multiple categories, separate them using semicolons, in the following format: `category1;category2;category`*n* For example: `ALARM IRP V1.0.0; CS IRP V1.0.0` The default is `" "` (the probe subscribes to all available notification categories).
NotificationFilter *string*	`-notificationfilter` *string*	Use this property to specify the filter that the notification IRP uses to limit the notifications sent to the probe. The default is `" "`.
NotificationIRPIOR File *string*	`-notificationirpiorfile` *string*	Use this property to specify the notification IRP IOR file. The default is `" "`.
NotificationIRPName *string*	`-notificationirpname` *string*	Use this property to specify the name used to resolve the Notification IRP. The default is `NotificationIRP`.

Table 5. Properties and command line options (continued)		
Property name	**Command line option**	**Description**
ORBCharEncoding *string*	`-orbcharencoding` *string*	Use this property to specify the native character encoding set used by the Object Request Broker (ORB) for character data. Specify a value of UTF8 for this property to enable the probe to receive events in languages that use multibyte characters, such as Chinese or Japanese. The default is `ISO-8859-1` (also known as Latin-1).
ORBDebug *string*	`-noorbdebug` (This is equivalent to **ORBDebug** with a value of `false`.) `-orbdebug` (This is equivalent to **ORBDebug** with a value of `true`.)	Use this property to specify whether the probe writes ORB debug messages. This property takes the following values: `false`: The probe does not write ORB debug messages to a log file. `true`: The probe writes ORB debug messages to the log file specified by the **ORBDebugFile** property. The default is `false`.
ORBDebugFile *string*	`-orbdebugfile` *string*	Use this property to specify the file location that the probe writes ORB debug messages. The default is `""`.
ORBLocalHost *string*	`-orblocalport` *string*	Use this property to specify the fully qualified domain name (FQDN) or IP address of the probe's host machine. The default is `" "`.
ORBLocalPort *integer*	`-orblocalport` *integer*	Use this property to specify the local port to which the Object Request Broker (ORB) listens. The default is `0` (the ORB selects an available port at random).
ORBWCharDefault *string*	`-orbwchardefault` *string*	Use this property to specify what wide character (wchar) set the IBM ORB uses when communicating with other ORBs that do not publish a wchar set. The default is UTF16.

Table 5. Properties and command line options (continued)		
Property name	**Command line option**	**Description**
Password *string*	`-password` *string*	Use this property to specify the password required to log in to the Huawei U2000 3GPP system. The default is `" "`. **Note :** If an encrypted password has been configured in the Huawei U2000 3GPP EMS for the RAN Sharing feature, you will have to obtain the encrypted password string from the EMS vendor, Huawei Technologies.
PersistentUnsubscribe *string*	`-nopersistentunsubscribe` (This is equivalent to **PersistentUnsubscribe** with a value of `false`.) `-persistentunsubscribe` (This is equivalent to **PersistentUnsubscribe** with a value of `true`.)	Use this property to enable persistent unsubscribe. `false`: Disables persistent unsubscribe. Probe will only unsubscribe the current notification session when shut down. `true`: Enables persistent unsubscribe. Probe will unsubscribe old notification session when attempting to start a new session. The default is `false`.
ResyncBatchSize *integer*	`-resyncbatchsize` *integer*	Use this property to specify the maximum number of alarms contained in each batch that the probe receives during a resynchronization operation. The minimum value for this property is 1. The default is `100`.
ResyncFilter *string*	`-resyncfilter` *string*	Use this property to specify the filter that the Alarm IRP uses to limit the alarms sent to the probe. The default is `""`. For more information about filtering alarms, see "Filtering notifications and alarms" on page 6.
SecondaryAlarmIRPIORFile *string*	`-secondaryalarmirpiorfile` *string*	Use this property to specify the secondary Alarm IRP IOR file for rotate endpoint (EMS) if **EnableFailover** is enabled. The default is `""`.

Table 5. Properties and command line options (continued)		
Property name	**Command line option**	**Description**
SecondaryAlarmIRP Name *string*	`-secondaryalarmirpname` *string*	Use this property to specify the name that the probe uses to resolve the secondary AlarmIRP point for rotate endpoint (EMS) if **EnableFailover** is enabled. The default is `""`.
SecondaryEntry PointIORFile *string*	`-secondaryepiorfile` *string*	Use this property to specify the location of the secondary Entry Point IRP IOR file for rotate endpoint (EMS) if **EnableFailover** is enabled. The default is `""`.
SecondaryNaming ServiceHost *string*	`-secondarynshost` *string*	Use this property to specify the secondary Naming Service host for rotate endpoint (EMS) if **EnableFailover** is enabled. The default is `""`.
SecondaryNaming ServiceIORFile *string*	`-secondarynsiorfile` *string*	Use this property to specify the secondary Naming Service IOR file for rotate endpoint (EMS) if **EnableFailover** is enabled. The default is `""`.
SecondaryNaming ServicePort *integer*	`-secondarynsport` *integer*	Use this property to specify secondary naming service port for rotate endpoint (EMS) if **EnableFailover** is enabled. The default is `0`.
SecondaryNotification IRPIORFile *string*	`-secondarynotifirpiorfile` *string*	Use this property to specify the secondary notification IRP IOR file for rotate endpoint (EMS) if **EnableFailover** is enabled. The default is `""`.
SecondaryNotification IRPName *string*	`-secondarynotifirpname` *string*	Use this property to specify the name used to resolve the secondary Notification IRP interface for rotate endpoint (EMS) if **EnableFailover** is enabled. The default is `""`.
SecurityProtocol *string*	`-securityprotocol` *string*	Use this property to specify the security protocol. The default is TLSv1.

Table 5. Properties and command line options (continued)		
Property name	**Command line option**	**Description**
TimeTick *integer*	`-timetick` *integer*	Use this property to specify the duration (in minutes) of the subscription to the target. The default is `-1` (this ensures that the session remains open indefinitely). **Note :** A value of 15 releases unused subscriptions and refreshes the subscription.
Username *integer*	`-username` *string*	Use this property to specify the username required to log in to the probe system. The default is `""`.

Properties and command line options provided by the Java Probe Integration Library (probe-sdk-java) version 11.0

All probes can be configured by a combination of generic properties and properties specific to the probe.

The following table describes the properties and command line options that are provided by the Java Probe Integration Library (probe-sdk-java) version 11.0.

Note : Some of the properties listed may not be applicable to your probe.

Table 6. Properties and command line options		
Property name	**Command line option**	**Description**
CommandPort *integer*	`-commandport` *integer*	Use this property to specify the port to which users can Telnet to communicate with the probe using the Command Line Interface (CLI) supplied. The default is `6970`.
CommandPortLimit *integer*	`-commandportlimit` *integer*	Use this property to specify the maximum number of Telnet connections that can be made to the probe. The default is `10`.
DataBackupFile *string*	`-databackupfile` *string*	Use this property to specify the path to the file that stores data between probe sessions. The default is `""`. **Note :** Specify the path relative to `$OMNIHOME/var`.

Table 6. Properties and command line options (continued)		
Property name	**Command line option**	**Description**
HeartbeatInterval *integer*	`-heartbeatinterval` *integer*	Use this property to specify the frequency (in seconds) with which the probe checks the status of the host server. The default is `1`.
Inactivity *integer*	`-inactivity` *integer*	Use this property to specify the length of time (in seconds) that the probe allows the port to receive no incoming data before disconnecting. The default is `0` (which instructs the probe to not disconnect during periods of inactivity).
InactivityAction *string*	`-inactivityaction` *string*	Use this property to specify the action the probe takes when the inactivity timeout is reached: `SHUTDOWN`: The probe sends a ProbeWatch message to notify the user and then shuts down. `CONTINUE`: The probe sends a ProbeWatch message to notify the user, but does not shut down. The default is `SHUTDOWN`.
InitialResync *string*	`-initialresync` *string*	Use this property to specify whether the probe performs resynchronization on startup. This property takes the following values: `false`: The probe does not request resynchronization on startup. `true`: The probe requests resynchronization on startup. For most probes, the default value for this property is `false`. If you are running the JDBC Probe, the default value for the **InitialResync** property is `true`. This is because the JDBC Probe only acquires data using the resynchronization process.

Table 6. Properties and command line options (continued)

Property name	Command line option	Description
MaxEventQueueSize *integer*	`-maxeventqueuesize` *integer*	Use this property to specify the maximum number of events that can be queued between the non native process and the ObjectServer. The default is `0`. **Note :** You can increase this number to increase the event throughput when a large number of events is generated.
ResyncInterval *integer*	`-resyncinterval` *integer*	Use this property to specify the interval (in seconds) at which the probe makes successive resynchronization requests. For most probes, the default value for this property is `0` (which instructs the probe to not make successive resynchronization requests). If you are running the JDBC Probe, the default value for the **ResyncInterval** property is `60`. This is because the JDBC Probe only acquires data using the resynchronization process.
RetryCount *integer*	`-retrycount` *integer*	Use this property to specify how many times the probe attempts to retry a connection before shutting down. The default is `0` (which instructs the probe to not retry the connection).
RetryInterval *integer*	`-retryinterval` *integer*	Use this property to specify the length of time (in seconds) that the probe waits between successive connection attempts to the target system. The default is `0` (which instructs the probe to use an exponentially increasing period between successive connection attempts, for example, the probe will wait for 1 second, then 2 seconds, then 4 seconds, and so forth).

Table 6. Properties and command line options (continued)		
Property name	**Command line option**	**Description**
RotateEndpoint *string*	`-rotateendpoint` *string*	Use this property to specify whether the probe attempts to connect to another endpoint if the connection to the first endpoint fails. This property takes the following values: `false`: The probe does not attempt to connect to another endpoint if the connection to the first endpoint fails. `true`: The probe attempts to connect to another endpoint if the connection to the first endpoint fails. The default is `false`.

Elements

The probe breaks event data down into tokens and parses them into elements. Elements are used to assign values to ObjectServer fields; the field values contain the event details in a form that the ObjectServer understands.

The following table describes the elements that the probe generates. Not all the elements described are generated for each event. The elements that the probe generates depend on the event type.

Table 7. Elements	
Element name	**Element description**
`$ClearSystemId`	This element identifies the system where the alarms in the IRP Manager are cleared.
`$ClearUserId`	This element contains the name of the user who cleared an alarm.
`$AckState`	This element specifies the acknowledgement state of the alarm.
`$AckSystemId`	This element specifies the system ID of the IRP Manager processing the notification.
`$AckTime`	This element specifies the time at which the user acknowledged the alarm.
`$AckUserId`	This element specifies the last user who has changed the acknowledgement state.
`$AdditionalText`	This element specifies information about the network element from which the alarm originated.
`$AlarmId`	This element specifies the identification information of the alarm as it appears in the alarm list.

Table 7. Elements (continued)

Element name	Element description
$BackupObject	This element specifies the distinguished Name (DN) of the backup object.
$BackupUpStatus	This element specifies whether the object has been backed up.
$Comments	This element contains comments about an alarm.
$CorrelatedNotifications	This element specifies the set of notifications to which this notification is considered to be correlated. This element is generated dynamically and its content is dependent on the IRPAgent.
$EventTime	This element specifies the time at which the event occurred.
$ManagedObjectClass	This element shows the managed object class of the network resource.
$ManagedObjectInstance	This element specifies the managed object instance of the network resource.
$MonitoredAttributes	This element contains the managed object attributes of the network resource.
$NotificationID	This element specifies the identification information of the notification.
$PerceivedSeverity	This element specifies the relative level of urgency for operator attention.
$ProbableCause	This element specifies further information about the probable cause of the alarm.
$ProposedRepairActions	This element specifies the proposed repair actions associated with the notification.
$Reason	This element indicates the reason that triggered the proposed repair action.
$SecurityAlarmDetector	This element indicates the security alarm detector for the device.
$ServiceProvider	This element contains the name of the service provider.
$ServiceUser	This element contains the name of the service user whose request for service led to the generation of a security alarm.

Table 7. Elements (continued)	
Element name	**Element description**
$SpecificProblem	This element specifies further information about the problem to which the notification relates.
$StateChangeDefinition	This element contains information about the state change.
$SystemDN	This element specifies the distinguished name (DN) used to identify the system.
$ThresholdInfo	This element specifies information about a threshold that has been crossed.
$TrendIndication	This element specifies how an observed condition has changed.
$VendorSpecificAlarmType	This element indicates the alarm type specific to the vendor.
$AlarmListAllignmentRequirement	This element indicates whether or not the alarm list requires alignment.
$AlarmRaisedTime	This element specifies the time at which the event was raised. The date is displayed in the format mm/dd/yy.
$AlarmClearedTime	This element specifies the time at which the event was cleared. The date is displayed in the format mm/dd/yy.
$AlarmChangedTime	This element specifies the time at which the event was changed. The date is displayed in the format mm/dd/yy.
$AlarmSpecialStatusId	This element specifies the special status ID of this alarm.
$RootCauseIndicator	This element indicates that this AlarmInformation is the root cause of the events captured by the notifications whose identifiers are in the related CorrelatedNotification instance.

Parsing the AdditionalText element

The Huawei U2000 3GPP system appends any data that is not in a name-value pair format to the AdditionalText element.

Earlier versions of the probe tokenized this unformatted data as an element name without a corresponding value, so this data could not be parsed by the rules file. The new element FULL_NV_ADDITIONAL_TEXT enables the probe to tokenize the contents of the AdditionalText element.

Example tokenization

This example shows the contents of an `AdditionalText` element parsed by the probe to create a `FULL_NV_ADDITIONAL_TEXT` token.

The `AdditionalText` element received by the probe:

```
AdditionalText: 'NeType: CG| NeLocation: | vendor: | neName: CG_22|
alarmName: No GSN Response| alarmLocation: igwb1: Alarm
parameter=0,1,172.18.154.59,10601| appendInfo: | ALARM 63023 Cleared Major CG 59
Software System
Alarm name = No GSN Response
Alarm raised time = 2011-07-25 02:33:49 DST
Location info = igwb1: Alarm parameter=0,1,172.18.154.59,10601
Cleared type = Cleared
Cleared time = 2011-07-25 02:34:23 DST'
```

The `FULL_NV_ADDITIONAL_TEXT` token created by the probe:

```
FULL_NV_ADDITIONAL_TEXT: NeType: CG| NeLocation: | vendor: | neName: CG_22|
alarmName: No GSN Response| alarmLocation: igwb1: Alarm
parameter=0,1,172.18.154.59,10601| appendInfo: | ALARM 63023 Cleared Major CG 59
Software System
Alarm name = No GSN Response
Alarm raised time = 2011-07-25 02:33:49 DST
Location info = igwb1: Alarm parameter=0,1,172.18.154.59,10601
Cleared type = Cleared
Cleared time = 2011-07-25 02:34:23 DST

NeType: CG

NeLocation:

vendor:

neName: CG_22

alarmName: No GSN Response

alarmLocation: igwb1: Alarm
parameter=0,1,172.18.154.59,10601

appendInfo:

ALARM 63023 Cleared Major CG 59
Software System
Alarm name = No GSN Response
Alarm raised time = 2011-07-25 02: 33:49 DST
Location info = igwb1: Alarm parameter=0,1,172.18.154.59,10601
Cleared type = Cleared
Cleared time = 2011-07-25 02:34:23 DST
```

Error messages

Error messages provide information about problems that occur while running the probe. You can use the information that they contain to resolve such problems.

The following table describes the error messages specific to this probe. For information about generic Netcool/OMNIbus error messages, see the *IBM Tivoli Netcool/OMNIbus Probe and Gateway Guide*.

Table 8. Error Messages

Error message	Description	Action
`NetcoolIRPManager: Exception when retrieving Alarm IRP`	Exception caught when probe was retrieving Alarm IRP information.	Check that the **`AlarmIRPName`** property is correctly set in the properties file.

Table 8. Error Messages (continued)

Error message	Description	Action
`NetcoolIRPManager: Failed to acknowledge_alarms`	The probe has failed to acknowledge the alarms.	Check that the NV_ALARM_ID specified by the CLI is correct.
`NetcoolIRPManager: Failed to unacknowledge_alarms`	The probe has failed to unacknowledge the alarms.	Check that the NV_ALARM_ID specified by the CLI is correct.
`NetcoolIRPManager: Failed to clear alarms`	The probe has failed to clear the alarms.	Check that the NV_ALARM_ID specified by the CLI is correct.
`NetcoolIRPManager: Failed to comment alarms`	The probe has failed to comment the alarms.	Check that the NV_ALARM_ID and comment text specified by the CLI is correct.
`Failed to retrieve probe specific properties`	The probe has failed to retrieve probe specific properties because of a service exception.	Service exception errors originate from the OIDK framework. Contact IBM support.
`Login: Failed to discover interface for`	Failed to discover IRP interface with specific IRP version when probe trying to login into EntryPoint IRP.	Check that **`EntryPointIORFile`**, **`AlarmIRPName`**, **`NotificationIRPName`** and **`IrpVersion`** properties are correctly set in the properties file.
`Failed to narrow Entry Point IRP interface`	Failed to narrow the object reference to the Entry Point IRP Agent object.	Check that the **`EntryPointIORFile`**, **`AlarmIRPName`**, and **`NotificationIRPName`** properties are correctly set in the properties file.
`Failed to retrieve EPIRP version`	Failed to narrow the object reference to the Entry Point IRP version.	Check that the **`EntryPointIORFile`**, **`AlarmIRPName`**, and **`NotificationIRPName`** properties are correctly set in the properties file.
`Name is null, cannot create element`	The probe was unable to parse the event because of its name; or an attribute name is not defined correctly in the IDL attribute name mapping.	Check if the IDL attribute name mapping is defined correctly with the IDL set used.
`Failed to parse element` `Bad operation exception during event parsing`	A problem occurred while parsing an element. The probe may not have parsed the event completely.	Error during parsing of the element in the probe. Contact IBM support.

Table 8. Error Messages (continued)		
Error message	**Description**	**Action**
`Failed to convert additional text using specified encoding standard`	The probe has failed to convert additional text using the specified encoding standard.	Check the **`EncodingStandard`** property is correctly set in the properties file. Ensure the incoming data (additional text) is in the supported encoding.
`Problem while setting the AdditionalText token`	The probe was unable to set the AdditionalText token.	Contact IBM support.
`Exception when retrieving Notification IRP`	The probe has encountered a problem when retrieving the Notification IRP version.	Check that the **`NotificationIRPName`** property is correctly set in the properties file. If using other connection options (for example, naming service), check the related properties.
`Notification subscribe: Error while trying to get subsription id and detach the notificationIRP`	The probe has failed to retrieve the subscription ID and detach the previous subscribed session.	Check if the session has been removed in EMS, or there is a connection issue to the EMS (ensure the EMS is active and running).
`Notification subscribe: Error while trying to attach the notificationIRP`	The probe was unable to attach or subscribe to the Notification IRP.	Check if the EMS is active and running.
`Detaching Subscription Error`	The probe was unable to detach subscription during unsubscribe.	Check if the session has been removed in EMS, or there is a connection issue to the EMS (ensure the EMS is active and running).

ProbeWatch messages

During normal operations, the probe generates ProbeWatch messages and sends them to the ObjectServer. These messages tell the ObjectServer how the probe is running.

The following table describes the ProbeWatch messages that the probe generates. For information about generic Netcool/OMNIbus ProbeWatch messages, see the *IBM Tivoli Netcool/OMNIbus Probe and Gateway Guide*.

Table 9. ProbeWatch messages		
ProbeWatch messages	**Description**	**Triggers/causes**
`END SYNCHRONIZATION`	The resynchronization of alarms has stopped.	The probe has resynchronized all the available alarms in the server.

Table 9. ProbeWatch messages (continued)		
ProbeWatch messages	**Description**	**Triggers/causes**
START SYNCHRONIZATION	The resynchronization of alarms has started.	The probe started to resynchronize the alarms collected in the system during the timeout period.

Running the probe

You run the probe from the command line.

To start the probe, use the following command:

```
$OMNIHOME/probes/nco_p_huawei_u2000_3gpp_corba
```

Known issues

At the time of release, a known issue was reported that you should be aware of when running the probe.

Alarm comments logged as user ID

The **userid_comment_alarm** CLI command enables you to add comments to an alarm by specifying the alarm ID (NV_ALARM_ID), the user ID of the user making the comment, and the text of the comment. It is also possible to use this command without specifying the user ID, in which case the user ID specified by the **CommentUserId** property is added to the related entry in the probe's log file.

However, when you use the command without specifying a user ID, and the comment that you add contains more than one word, the entire comment string is logged in the probe's log file in place of the user ID. This happens regardless of the value specified by the **CommentUserId** property.

The following example demonstrates how the CLI successfully completes a **userid_comment_alarm** command where only the alarm ID and the comment string are entered:

```
> userid_comment_alarm 2042480408 testing purposes only
Completed userid_comment_alarm
```

In this case, the related entry in the probe's log file contains the comment string `testing purposes only` in place of the user ID.

Command line input no whitespace limitation

The **resynch_filter** CLI command enables you to perform partial resynchronization with the CORBA interface using the *filter* parameter. However, currently the command port does not allow the *filter* input parameter to contain any whitespaces in between characters.

The following example demonstrates how the CLI successfully completes a **resynch_filter** command without any whitespaces:

```
resynch_filter $h==3
```

A command containing whitespace will be unsuccessful, for example `resynch_filter $h == 3`.

Multibyte characters not supported in the AdditionalText field

Currently the probe does not support the proper display of multibyte characters (that is, characters encoded in UTF-8) in the `AdditionalText` fields.

Probe release ID not printed

On Netcool/OMNIbus 8.1 on AIX, the probe's release ID is not printed when starting the probe using –`version`.

To get the probe's version information, start the probe with DEBUG level logging and get the release ID from the log file header.

Chapter 2. Migrating from probe for Huawei M2000 (CORBA)

The Probe for Huawei U2000 3GPP (CORBA) replaces the Probe for Huawei M2000 (CORBA).

This chapter contains guidance on how to migrate from the Probe for Huawei M2000 (CORBA) to the Probe for Huawei U2000 3GPP (CORBA). The migration procedure has the following stages:

1. Review the differences between probes and identify any gaps in terms of functionality, properties, elements and rules file.
2. Install the new probe.
3. Migrate the properties file.
4. Customize the rules file.
5. Run and test the new probe.
6. Optimize property values and the rules file.

Note : Where possible, carry out the migration in a test environment or a simulation of the production environment so that the work does not interfere with the production environment. Change over to using the Probe for Huawei U2000 3GPP (CORBA) in production once you are sure that it behaves in the same way as the Probe for Huawei M2000 (CORBA).

This guide contains the following sections:

Comparison of probe features

Both probes have some features in common, and additional features specific to the Probe for Huawei U2000 3GPP (CORBA).

Common features

The following features are common to both probes:

Table 10. Features common to both probes

Functional category	Features
Connecting to the CORBA interface	Connect through an IOR file. Connect through a Naming Service host and port. Connect through a Naming Service IOR file.
Resynchronization	Retrieve alarms from the system on startup before receiving new alarms. Retrieve alarms using a severity filter. Retrieve alarms in batches.

Table 10. Features common to both probes (continued)	
Functional category	**Features**
Data acquisition	Authentication with the NMS or EMS using a username and password. Ability to receive alarms and notifications. Heartbeat status check. Inactivity timeout. Reconnection and probe backoff. Support for Unicode and non-Unicode characters.

Features specific to the Probe for Huawei U2000 3GPP (CORBA)

The Probe for Huawei U2000 3GPP (CORBA) has the following additional features that are not present in the Probe for Huawei M2000 (CORBA):

Table 11. Features specify to Probe for Huawei U2000 3GPP (CORBA)	
Functional category	**Property Name**
SSL connectivity	**EnableSSL** **KeyStore** **KeyStorePassword** **SecurityProtocol**
EMS redundancy	**EnableFailover** **SecondaryAlarmIRPIORFile** **SecondaryAlarmIRPName** **SecondaryEntryPointIORFile** **SecondaryNamingServiceHost** **SecondaryNamingServiceIORFile** **SecondaryNamingServicePort** **SecondaryNotificationIRPIORFile** **SecondaryNotificationIRPName**
Persistence unsubscribe	**PersistentUnsubscribe**
Alarm IRP IOR file	**AlarmIRPIORFile**
Notification IRP IOR file	**NotificationIRPIORFile**
Naming service host/port	**NamingServiceHost** **NamingServicePort**
Naming service IOR file	**NamingServiceIORFile**
NSProbe max queue size	**MaxEventQueueSize**

Table 11. Features specify to Probe for Huawei U2000 3GPP (CORBA) (continued)	
Functional category	**Property Name**
Resync with interval	**ResyncInterval**

Features not available in the Probe for Huawei U2000 3GPP (CORBA)

The Probe for Huawei U2000 3GPP (CORBA) has a number of features that are not currently available:

- FTP Password

Before deploying the Probe for Huawei U2000 3GPP (CORBA) ensure that you do not require any of these features.

Migration procedure

Use this procedure to replace the Probe for Huawei M2000 (CORBA) with the Probe for Huawei U2000 3GPP (CORBA).

Determining probe differences

Probe for Huawei U2000 3GPP (CORBA) includes new functionality and has dropped certain features and properties.

Use the information in "Comparison of probe features" on page 33 to determine the features that are available in the Probe for Huawei U2000 3GPP (CORBA).

Installing the Probe for Huawei U2000 3GPP (CORBA)

Follow the advice in "Installing probes" on page 2 to download and install the probe in to a test environment.

Migrating properties

Determine the values required for the properties file of the Probe for Huawei U2000 3GPP (CORBA). Use the properties file from the Probe for Huawei M2000 (CORBA) to set the correct values.

Note : The Probe for Huawei U2000 3GPP (CORBA) uses different names for some properties to those used in the Probe for Huawei M2000 (CORBA) as the following table shows:

Table 12. New property names	
Huawei M2000 (CORBA)	**Huawei U2000 3GPP (CORBA)**
AlarmIrpName	**AlarmIRPName**
NotificationIrpName	**NotificationIRPName**
EntryPointIrpFile	**EntryPointIORFile**
AgentHeartbeat	**HeatbeatInterval**

Table 12. New property names (continued)	
Huawei M2000 (CORBA)	**Huawei U2000 3GPP (CORBA)**
AlarmFilter	**ResyncFilter**
Retry	**RetryCount** **RetryInterval**
Resynch	**InitialResync** **ResyncInterval**
Timeout	**Inactivity**

Customizing the rules file

Edit the rules file for the Probe for Huawei U2000 3GPP (CORBA) to:

- Migrate custom rules from the Probe for Huawei M2000 (CORBA) rules file to the Probe for Huawei U2000 3GPP (CORBA) rules file.
- Apply changes to the @ClassID, @Manager, and lookup tables as required.

Note : The Probe for Huawei U2000 3GPP (CORBA) may not be able to parse certain attributes if the vendor does not follow the probe standard or has implemented their own types that are not probe compliant.

Attributes

There are some differences in the names or values of attributes between the two probes. The following table indicates these differences. Be sure to make the necessary changes if you copy over rules from the legacy rules file.

Table 13. Differences in rules file attributes	
Huawei M2000 (CORBA)	**Huawei U2000 3GPP (CORBA)**
$EVENT_NAME	$EventName
$EVENT_TYPE	$EventType
$domain_name	$DomainName
$NV_ALARM_ID	$AlarmId
$NV_SYSTEM_DN	$SystemDN
$NV_MANAGED_OBJECT_INSTANCE	$ManagedObjectInstance
$NV_ACK_STATE	$AckState
$NV_PROBABLE_CAUSE	$ProbableCause
$NV_PERCEIVED_SEVERITY	$PerceivedSeverity
$NV_SPECIFIC_PROBLEM	$SpecificProblem
$NV_TREND_INDICATION	$TrendIndication
$NV_THRESHOLD_INFO	$ThresholdInfo
$NV_EVENT_TIME	$EventTime
$NV_COMMENTS	$Comments
$NV_NOTIFICATION_ID	$NotificationID

Running and testing the probe

Run the probe and ensure it is communicating with the NMS or EMS correctly.

To run and test the probe:

1. Start the probe from the command line, specifying the minimum message level of debug and that an initial resynchronization is to occur. For example:

   ```
   $OMNIHOME/porbes/nco_p_huawei_u2000_3gpp_corba -messagelog stdout -
   messagelevel debug
   ```

2. Ensure that the probe connects to the target system successfully. Look for the following message in the probe's log file:

   ```
   2015-06-03T16:09:09: Information: I-JPR-000-000: Probe connected
   ```

   If the probe fails to connect:

   - Check and adjust the properties related to setting up a connection. See “Connecting to the CORBA interface” on page 5 for information on the connection properties and how to set them.
   - Ensure that any firewall between the probe host and the NMS or EMS is configured to allow traffic to pass from one end to the other in both directions.

3. Check that the probe successfully synchronizes with the NMS or EMS. Look for messages similar to the following in the probe's log file:

   ```
   Information: I-JPR-000-000: Resynchronizing Probe
   Information: I-UNK-000-000: Probewatch: START SYNCHRONIZATION
   Debug: D-JPR-000-000: Send request for active alarms
   Debug: D-UNK-000-000: (Non-encrypted property) ResyncFilter->
   Debug: D-JPR-000-000:  Performing full resynch with Alarmfilter property :
   Debug: D-JPR-000-000:  calling get_alarm_list()
   Debug: D-JPR-000-000: Statistic of alarms received in one batch
   Debug: D-JPR-000-000: ResyncAlarmData [isAllAlarm=true, criticalCount=0,
   majorCount=0, minorCount=0, warningCount=0, indeterminateCount=0, clearedCount=0]
   Debug: D-JPR-000-000: Parsing alarm
   Information: I-UNK-000-000: Probewatch: END SYNCHRONIZATION
   ```

   Troubleshoot any synchronization errors, including the values of the synchronization properties. See “Filtering notifications and alarms” on page 6 for information on synchronization.

4. Check that the probe correctly parses alarms with the Event Processor. Check for any unsupported types for event parsing.
5. Check the log file for errors in event parsing or unsupported types.

   Check also for attributes having a null value or one that shows as `'UNKNOWN'`.

6. Check that events appear in the Event List and that they contain the expected elements and values.

   Modify the rules file if the values in the Event List do not meet your requirements.

Optimizing property values and the rules file

As a result of testing the probe, make any changes and optimizations necessary to the properties file and the rules file. Then test the probe again. Repeat this process until the probe behaves correctly and the Event List contains all the expected events with all the required elements and values.

Appendix A. Notices and Trademarks

This appendix contains the following sections:

- Notices
- Trademarks

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing 2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who want to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Software Interoperability Coordinator, Department 49XA

3605 Highway 52 N
Rochester, MN 55901
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM, the IBM logo, ibm.com, AIX, Tivoli, zSeries, and Netcool are trademarks of International Business Machines Corporation in the United States, other countries, or both.

Adobe, Acrobat, Portable Document Format (PDF), PostScript, and all Adobe-based trademarks are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, other countries, or both.

Intel, Intel Inside (logos), MMX, and Pentium are trademarks of Intel Corporation in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

SC27-6583-03