IBM® Tivoli® Netcool/OMNIbus Generic Probe for the 3GPP Interface (CORBA)
3.0

Reference Guide
November 23, 2017

SC27-6561-02

Notice

Before using this information and the product it supports, read the information in Appendix A, “Notices and Trademarks,” on page 35.

Edition notice

This edition (SC27-6561-02) applies to version 3.0 of IBM Tivoli Netcool/OMNIbus Generic Probe for the 3GPP Interface and to all subsequent releases and modifications until otherwise indicated in new editions.

This edition replaces SC27-6561-01.

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 Generic Probe for the 3GPP Interface documentation is provided in softcopy format only. To obtain the most recent version, visit the IBM Tivoli Netcool Knowledge Center:

http://www-01.ibm.com/support/knowledgecenter/#!/SSSHTQ/omnibus/common/kc_welcome-444.html

Table 1. Document modification history

Document version	Publication date	Comments
SC27-6561-00	March 12, 2015	First IBM publication.
SC27-6561-01	August 6, 2015	The probe now supports versions 3.2, 5.5.1, 6.3, and 6.4 of the 3GPP interface. Support for secure SSL connectivity added. "Summary" on page 2 updated. "Configuring the probe" on page 4 updated. "SSL-based connectivity" on page 4 added. "Properties and command line options" on page 15 updated. **Enhancements:** Version 2 of the Generic Probe for the 3GPP Interface (CORBA) includes the following enhancements: • **RFE 64647**: Support for SSL-based connectivity added.
SC27-6561-02	November 23, 2017	The probe now supports versions 7.0 and 9.1 of the 3GPP interface. **APARs:** Version 3 of the Generic Probe for the 3GPP Interface (CORBA) address the following APARs: • **IT16770**: Enable probe to resync when alarms are returned in batches. • **IT11805**: Support `AckTime` and `EventTime` parsing on 3GPP 3.2. **Enhancements:** Version 3 of the Generic Probe for the 3GPP Interface (CORBA) includes the following enhancements: • **RFE 94030**: Support added for 3GPP interface version 7.0 • **RFE 50037**: Support added for 3GPP interface version 9.1

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`
Red Hat Linux® and SUSE systems	`linux2x86`
Linux for System z	`linux2s390`
Solaris systems	`solaris2`
Windows systems	`win32`

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. Generic 3GPP Probe

The IBM Tivoli Netcool/OMNIbus Generic Probe for the 3GPP Interface monitors devices that manage 3G telecommunication networks compliant with 3GPP standards and use a CORBA interface.

Version 3.2

For version 3.2 of the 3GPP interface, the probe complies with the following 3GPP standards:

- 32.111-3 V3.2.0 - Alarm IRP
- 32.303 V3.2.0 - Notification IRP

Version 5.5.1

For version 5.5.1 of the 3GPP interface, the probe complies with the following 3GPP standards:

- 32.111-3 V5.5.1 - Alarm IRP
- 32.303 V5.2.0 - Notification IRP
- 32.323 V5.2.0 - Generic Network

Version 6.3

For version 6.3 of the 3GPP interface, the probe complies with the following 3GPP standards:

- 32.111-3 V6.3.0 - Alarm IRP
- 32.303 V6.3.0 - Notification IRP
- 32.363 V6.3.0 - Entry Point IRP

Version 6.4

For version 6.4 of the 3GPP interface, the probe complies with the following 3GPP standards:

- 32.111-3 V6.4.0 - Alarm IRP
- 32.303 V6.3.0 - Notification IRP
- 32.363 V6.3.0 - Entry Point IRP

Version 7.0

For version 7.0 of the 3GPP interface, the probe complies with the following 3GPP standards:

- 32.111-3 V7.0.0 - Alarm IRP
- 32.303 V7.0.0 - Notification IRP
- 32.363 V7.0.0 - Entry Point IRP

Version 9.1

For version 9.1 of the 3GPP interface, the probe complies with the following 3GPP standards:

- 32.111-3 V9.1.0 - Alarm IRP
- 32.303 V9.0.0 - Notification IRP
- 32.363 V9.0.0 - Entry Point IRP

The following topics describe the probe and how it works:

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	3G network devices that comply with 3GPP standards versions 3.2, 5.5.1, 6.3, 6.4, 7.0, and 9.1.
Probe executable name	`nco_p_generic_3gpp`
Package Version	3.0
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=swg21697989
Properties file	`$OMNIHOME/probes/`*`arch`*`/generic_3gpp.props`
Rules file	`$OMNIHOME/probes/`*`arch`*`/generic_3gpp.rules`
Minimum 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 a CORBA interface.
Multicultural support	Not 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.

Configuring the probe

To configure the probe prior to running it, you must ensure the installation requirements are met. You must also update the rules file using probe-specific information.

Updating the rules file

The probe is supplied with the following rules file:

- `generic_3gpp.rules`

The probe is also supplied with the following lookup table:

- `generic_3gpp.lookup`

This file is installed in the following location: `$OMNIHOME/probes/includes/`

It is referenced in the rules file by the following command:

```
include "../includes/generic_3gpp.lookup"
```

Note : `$OMNIHOME` cannot be used in the paths to the lookup files. You must enter the full path to the IBM Tivoli Netcool/OMNIbus installation directory.

SSL-based connectivity

The Generic Probe for the 3GPP Interface (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. 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 vendor.

Creating the SSL keystore

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 3GPP interface, follow these steps:

1. Set the **EnableSSL** property to `true`.

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

   - **KeyStore**
   - **KeyStorePassword**
   - **SecurityProtocol**

2. Use the **KeyStore** property to specify the location of the keystore file *`trusted_keystore`*`.jks`.
3. Use the **KeyStorePassword** property to specify a password for the keystore.
4. Encrypt the keystore file password using the `nco_g_crypt` utility.

Data acquisition

The probe connects to the target system through a Common Object Request Broker Architecture (CORBA) interface. CORBA is an Object Management Group specification that provides a standard interface definition between objects in a distributed environment; that is, it allows applications to communicate with one another regardless of where they are located or who has designed them.

On startup, the probe initializes an ORB and connects to the target system's Alarm IRP objects. The probe then resynchronizes with the target system's Element Manager and acquires the alarms events currently stored in the target system's Element Manager.

The probe then processes the acquired alarms, setting most attributes as tokens, and generates an AckAlarmID token. These tokens are sent to the ObjectServer as events. Once the process is complete, the probe subscribes to the online events, processes them, and then forwards them to the ObjectServer.

The probe checks the status of the IRP agent every 60 seconds. You can change this frequency if required using the **Agentheartbeat** property.

The following topics describe how the probe acquires data:

- "Device connections through the CORBA interface" on page 6
- "Retrieving objects" on page 7
- "Filters for notifications and alarms" on page 7
- "Command line interface" on page 8
- "Peer-to-peer failover functionality" on page 14

Device connections through the CORBA interface

The probe uses the CORBA interface to retrieve alerts from the monitored device. The probe can use one of two methods to connect to the device: Interoperable Object Reference (IOR) files or the Naming Service.

IOR files

If using IOR files, there are two methods for connecting to the endpoints:

Method 1:

Method 1 uses the following properties:

- **EntryPointIORFile**
- **AlarmIRPName**
- **NotificationIRPName**

The probe retrieves the object reference of the Entry Point object from the IOR file specified by the **EntryPointIORFile** property.

It then sends a resynchronization request to the Alarm IRP object specified by the **AlarmIRPName** property and sends an active alarms subscription request to the Notification IRP object specified by the and **NotificationIRPName** property.

Method 2:

Method 2 uses the following properties:

- **AlarmIRPIORFile**
- **NotificationIORFile**

The probe retrieves the object reference of the Alarm IRP object from the IOR file specified by the **AlarmIRPIORFile** property and the probe retrieves the object reference of the Notification IRP object from the IOR file that is specified by the **NotificationIRPIORFile** property.

Naming Service

If you are using the Naming Service, you must configure the following properties:

- **NamingServiceHost**
- **NamingServicePort**
- **NamingServiceIorFile**
- **AlarmIRPName**
- **NotificationIRPName**

If the IOR file properties are not specified, the probe retrieves the object references of the AlarmIRP object and NotificationIRP object from the Naming Service. To locate the Naming Service, the probe either uses the **NamingServiceHost** and **NamingServicePort** properties to identify the host name and port number of the Naming Service, or uses the IOR file specified by the **NamingServiceIorfile** property.

The Naming Service uses the values that are specified by the **AlarmIrpName** and **NotificationIrpName** properties to retrieve the object references to the IRP objects.

Retrieving objects

The probe initially receives a list of all active alarms from the AlarmIRPOperation server. The probe then connects to the NotificationIRPOperation server and uses the CORBA notification push model to receive new alarms from the server as they are generated.

Filters for notifications and alarms

The **NotificationFilter** and **AlarmFilter** properties allow you to specify what notifications and alarms are sent to the probe. When you use these properties, you must use the actual token names.

For example, the token h represents the element PerceivedSeverity. So, to specify that the probe is sent only notifications with a perceived severity of 3, you must set the **NotificationFilter** property to $h == 3.

You can specify more complex filters using AND and OR statements. For example, to specify that the probe is sent notifications with a perceived severity of 3 or 4, you must set the **NotificationFilter** property to $h == 3 or $h == 4.

To specify that the probe is only sent notifications for a specific managed element, set the **NotificationFilter** property to Managed_Node_Name~$e where $e represents the element ManagedObjectInstance and Managed_Node_Name is the name of the managed object.

For example, if the set of alarms that you require return a ManagedObjectInstance of SubNetwork=ONRM_RootMo, SubNetwork=SNMP, ManagedElement=SP1, set the **NotificationFilter** property to SP1'~$'f.

Note : The tilde character (~) is required because spaces cannot be entered in this property. For string comparisons, the first argument is considered to be contained in the second argument; which is why $f is listed second to the literal.

The following table displays the token mappings for use with the **AlarmFilter** and **NotificationFilter** properties.

Table 4. Token mappings

Element	Token
NotificationID	a
EventTime	b
SystemDN	c
ManagedObjectClass	d
ManagedObjectInstance	e
AlarmId	f
ProbableCause	g
PerceivedSeverity	h
SpecificProblem	i
AdditionalText	j
AckTime	k
AckUserId	l
AckSystemId	m
AckState	n
Comments	o

Table 4. Token mappings (continued)	
Element	**Token**
BackupUpStatus	p
BackupObject	q
ThresholdInfo	r
TrendIndication	s
StateChangeDefinition	t
MonitoredAttributes	u
ProposedRepairActions	v
CorrelatedNotifications	w
Reason	x
ClearUserId	y
ClearSystemId	z
AlarmListAllignmentRequirement	ff
ServiceUser	gg
ServiceProvider	hh
SecurityAlarmDetector	ii
VendorSpecificAlarmType	jj
AlarmRaisedTime	kk
AlarmClearedTime	ll

Command line interface

When using the probe with IBM Tivoli Netcool/OMNIbus V7.3.1 or earlier, there is a command line interface (CLI) that you can use to manage the probe over a Telnet connection. For IBM Tivoli Netcool/OMNIbus V7.4 and later, use the HTTP/HTTPS command interface.

To use the CLI, ensure the following probe properties have suitable values:

- **CommandPort**: Set this to the port number on the probe that Telnet connects through.
- **CommandPortLimit**: Set this to the maximum number of CLI connections that can be open concurrently.

Table 5. CLI commands	
Command	**Description**
acknowledge_alarm *alarm_id*	Use this command to acknowledge an alarm in the 3GPP interface. **Note :** This command takes as a parameter the AlarmId of the alarm being acknowledged. Only one alarm can be acknowledged at a time. This command also uses the values specified by the **AckSystemId** and **AckUserId** properties in the properties file.
exit	This command closes the Command Port connection.

Table 5. CLI commands (continued)	
Command	**Description**
help	Use this command to display online help about the CLI.
clear_alarm *alarm_id*	Use this command to clear an alarm in the 3GPP interface. **Note :** Version 3.2 of the 3GPP Interface does not support this command.
resynch_all	Use this command to perform a full resynchonization with the 3GPP interface.
resynch_filter *filter_name*	Use this command to perform partial resynchronization with the 3GPP interface.
unacknowledge_alarm *alarm_id*	Use this command to unacknowledge an alarm in the 3GPP interface.
userid_acknowledge_alarm *user_id alarm_id*	Use this command to acknowledge an alarm in the 3GPP interface by specifying the `AlarmId` of the alarm being acknowledged and the `AckUserId`. The format of the alarm is: *userID ID*.
userid_clear_alarm *user_id alarm_id*	Use this command to clear an alarm by specifying the identifier user (`AckUserId`) of the user who created the alarm. If you specify a value for the **ClearUserId** property, the *user_id* parameter is not required. **Note :** Version 3.2 of the 3GPP Interface does not support this command.
userid_unacknowledge_alarm *user_id alarm_id*	Use this command to unacknowledge an alarm in the 3GPP interface by specifying the `AlarmId` of the alarm being acknowledged and the `AckUserId`.
version	Use this command to display the version of the probe.

Note : Because the CLI is based upon Telnet connections, you can connect to the probe from anywhere. This means that simple scripts can be set up to allow users to acknowledge selected events from the event list by creating desktop tools to Telnet to the probe, send a command, and then close the connection.

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 version 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/generic_3gpp -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:

  acknowlege_alarm
  clear_alarm
  help
  resync_all
  resync_filter
  unacknowlege_alarm
  userid_acknowlege_alarm
  userid_clear_alarm
  userid_unacknowlege_alarm
  version

  Note : Version 3.2 of the 3GPP Interface does not support **clear_alarm** or **userid_clear_alarm**.
- *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`.

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:8080/probes/generic_3gpp -datatype
application/JSON -method POST -data '{"command":"version", "params":[]}'
```

acknowledge_alarm

Use the **`acknowledge_alarm`** command to acknowledge an alarm.

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

```
-data '{"command":"acknowledge_alarm", "params":[{"alarm_id":"alarmId"}]}'
```

Where *alarmId* is the identifier stored in the alarm's `AlarmId` field.

The following example acknowledges the alarm with an `AlarmId` of `50047933`:

```
$OMNIHOME/bin/nco_http -uri http://localhost:8080/probes/generic_3gpp -datatype
application/JSON -method POST -data '{"command":"acknowledge_alarm", "params":
[{"alarm_id":"50047933"}]}'
```

clear_alarm

Use the **`clear_alarm`** command to clear an alarm.

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

```
-data '{"command":"clear_alarm", "params":[{"alarm_id":"alarmId"}]}'
```

Where *alarmId* is the identifier stored in the alarm's `AlarmId` field.

The following example clears the alarm with an `AlarmId` of `50047933`:

```
$OMNIHOME/bin/nco_http -uri http://localhost:8080/probes/generic_3gpp -datatype
application/JSON -method POST -data '{"command":"clear_alarm", "params":
[{"alarm_id":"50047933"}]}'
```

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:8080/probes/generic_3gpp -datatype
application/JSON -method POST -data '{"command":"help", "params":[]}'
```

resync_all

Use the **`resync_all`** command to perform a complete resynchronization with the endpoint.

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

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

The following example resynchronizes the probe:

```
$OMNIHOME/bin/nco_http -uri http://localhost:8080/probes/generic_3gpp -datatype
application/JSON -method POST -data '{"command":"resync_all", "params":[]}'
```

resync_filter

Use the **resync_filter** command to perform a resynchronization using a custom filter.

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

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

Where *filter* limits the alarms that are resynchronized.

The following example resynchronizes alarms that have perceived severity of CRITICAL:

```
$OMNIHOME/bin/nco_http -uri http://localhost:8080/probes/generic_3gpp -datatype
application/JSON -method POST -data '{"command":"resync_filter", "params":
[{"resync_filter":"perceivedSeverity==CRITICAL"}]}'
```

unacknowledge_alarm

Use the **unacknowledge_alarm** command to unacknowledge an alarm.

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

```
-data '{"command":"unacknowledge_alarm", "params":[{"alarm_id":"alarmId"}]}'
```

Where *alarmId* is the identifier stored in the alarm's AlarmId field.

The following example unacknowledges the alarm with an AlarmId of 50047933:

```
$OMNIHOME/bin/nco_http -uri http://localhost:8080/probes/generic_3gpp -datatype
application/JSON -method POST -data '{"command":"unacknowledge_alarm",
"params":[{"alarm_id":"50047933"}]}'
```

userid_acknowledge_alarm

Use the **userid_acknowledge_alarm** command to acknowledge an alarm.

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

```
-data '{"command":"userid_acknowledge_alarm", "params":
[{"ack_user_id":"userId","alarm_id":"alarmId"}]}'
```

Where:

- *alarmId* is the identifier stored in the alarm's AlarmId field.
- *userId* is the user name of the user acknowledging the alarm.

The following example acknowledges the alarm with the following characteristics:

Alarm Identifier: 50047933
User ID: your_user_id

```
$OMNIHOME/bin/nco_http -uri http://localhost:8080/probes/generic_3gpp -datatype
application/JSON -method POST -data '{"command":"userid_acknowledge_alarm",
"params":[{"ack_user_id":"your_user_id","alarm_id":"50047933"}]}'
```

userid_clear_alarm

Use the **userid_clear_alarm** command to clear an alarm.

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

```
-data '{"command":"userid_clear_alarm", "params":
[{"clear_user_id":"userId","alarm_id":"alarmId"}]}'
```

Where:

- *alarmId* is the identifier stored in the alarm's AlarmId field.
- *userId* is the user name of the user acknowledging the alarm.

The following example clears the alarm with the following characteristics:

Alarm Identifier: 50047933
User ID: your_user_id

```
$OMNIHOME/bin/nco_http -uri http://localhost:8080/probes/generic_3gpp -datatype
application/JSON -method POST -data '{"command":"userid_clear_alarm", "params":
[{"clear_user_id":"your_user_id","alarm_id":"50047933"}]}'
```

userid_unacknowledge_alarm

Use the **userid_unacknowledge_alarm** command to unacknowledge an alarm.

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

```
-data '{"command":"userid_unacknowledge_alarm", "params":
[{"ack_user_id":"userId","alarm_id":"alarmId"}]}'
```

Where:

- *alarmId* is the identifier stored in the alarm's AlarmId field.
- *userId* is the user name of the user acknowledging the alarm.

The following example unacknowledgs the alarm with the following characteristics:

Alarm Identifier: 50047933
User ID: your_user_id

```
$OMNIHOME/bin/nco_http -uri http://localhost:8080/probes/generic_3gpp -datatype
application/JSON -method POST -data '{"command":"userid_unacknowledge_alarm",
"params":[{"ack_user_id":"your_user_id","alarm_id":"50047933"}]}'
```

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 **resync** command:

```
Information: I-UNK-000-000: NSProbeBidirCB: Thread id is 0x94d9008
{command:resync,params:[]}
Information: I-UNK-000-000: Probewatch: Starting the resynch of alarm list
Debug: D-UNK-000-000: Rules file processing took 28 usec.
Debug: D-UNK-000-000: Flushing events to object servers
Debug: D-UNK-000-000: Flushing events to object servers
Debug: D-JPR-000-000: com.ibm.tivoli.netcool.omnibus.probe.bidi.CommandHandler.
executeCommand ENTERING
Debug: D-JPR-000-000: com.ibm.tivoli.netcool.omnibus.probe.bidi.CommandHandler.
checkParams ENTERING
Debug: D-JPR-000-000: com.ibm.tivoli.netcool.omnibus.probe.bidi.CommandHandler.
checkParams EXITING
Debug: D-JPR-000-000: Send request for active alarms
Information: I-UNK-000-000: Probewatch: Finished the resynch of alarm list
```

These messages can also help to isolate problems with a command. For example, the following shows the log messages for an `unackAlarm` command that contained an invalid alarm identifier.

```
Information: I-UNK-000-000: NSProbeBidirCB: Thread id is
0x9ec8b48 {"command":"unackAlarm","params":[{"alarmId":"abcd","emsId":"EMS1",
"managedElementId":"ME1","username":"root"}]}
Debug: D-JPR-000-000: com.ibm.tivoli.netcool.omnibus.probe.
bidi.CommandHandler.executeCommand ENTERING
Debug: D-JPR-000-000: com.ibm.tivoli.netcool.omnibus.probe.
bidi.CommandHandler.checkParams ENTERING
Debug: D-JPR-000-000: com.ibm.tivoli.netcool.omnibus.probe.
bidi.CommandHandler.checkParams EXITING
Debug: D-JPR-000-000: Unacknowledge alarm with alarm ID: abcd on EMS:
and ME: ME1, and username: root
Information: I-JPR-000-000: There are : 1 alarms that failed to be unacknowledged.
```

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 : '{"command":"ackAlarm", "params":[{"alarmId":"alarm1",
"emsId":"EMS1", "managedElementId":"ME1", "username":"root"}]}'
DataType : 'application/JSON'
Method : 'POST'
```

To use this set of values use the following **nco_http** command:

```
$OMNIHOME/bin/nco_http -uri http://test1.example.com:6789
```

Peer-to-peer failover functionality

The probe 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 does not forward events to the ObjectServer. If the master probe shuts down, the slave probe stops receiving heartbeats from the master and any events it receives thereafter are forwarded to the ObjectServer on behalf of the master probe. When the master probe is running again, the slave probe continues to receive events, but no longer sends 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    :    6789 # [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    :    6789 # [communication port between master and slave probe]
Mode        :    "slave"
PidFile     : "slave_pid_file"
```

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 6. Properties and command line options

Property name	Command line option	Description
AckSystemId *string*	`-acksystemid` *string*	Use this property to specify the processing system on which the IRP Manager runs. This is used by the **acknowledge_alarm** CLI command. The default is " ".
AckUserId *string*	`-ackuserid` *string*	Use this property to specify the name of the user acknowledging the alarm. This is used by the **acknowledge_alarm** CLI command. The default is " ".
AlarmIRPIOR *string*	`-alarmirpior` *string*	Use this property to specify the alarm IRP object reference. The default is " ".

Table 6. Properties and command line options (continued)

Property name	Command line option	Description
AlarmIRPIORFile *string*	`-alarmirpiorfile` *string*	Use this property to specify the path to the Alarm IRP object reference. The default is `" "`.
AlarmIrpName *string*	`-alarmirp` *string*	Use this property to specify the name of the Alarm IRP object to which the probe sends a resynchronization request. The default is `AlarmIRP=1`.
ClearSystemId *string*	`-clearsystemid` *string*	Use this property to specify the system identifier of the alarms that the **system_clear_alarms** CLI command clears. The default is `" "`.
ClearUserId *string*	`-clearuserid` *string*	Use this property to specify the user identifier of the alarms that the **userid_clear_alarms** CLI command clears. The default is `" "`.
EnableSSL *string*	`-noenablessl` (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 3GPP interface is enabled or disabled. This property takes the following values: `false`: SSL connectivity between the probe and the 3GPP interface is disabled. `true`: SSL connectivity between the probe and the 3GPP interface is enabled. The default is `false`.
EntryPointIOR *string*	`-entrypointior` *string*	Use this property to specify the Entry Point object reference . The default is `" "`.
EntryPointIORFile *string*	`-entrypointiorfile` *string*	Use this property to specify the path to the Entry Point object reference file. The default is `" "`.
EntryPointIRPName *string*	`-entrypointirpname` *string*	Use this property to specify the name of the Entry Point IRP object. The default is `" "`.

Table 6. Properties and command line options (continued)

Property name	Command line option	Description
IDLAttrMapFile *string*	`-idlattrmapfile` *string*	Use this property to specify the IDD attribute map that the probe uses when creating readable tokens for the probe log and rules file. The default is `$OMNIHOME/probe/includes/generic_3gpp_v6_4_RuleElementMap.xml` You will need to change the value of this property to specify the full path to the file, for example: `opt/IBM/tivoli/netcool81/ omnibus/probe/includes/generic_3gpp_v6_4_RuleElementMap.xml`
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 specified by the **Keystore** property. The default is `""`. **Note :** You must encrypt this password using the `nco_g_crypt` utility with Netcool/OMNIbus.
NamingServiceHost *string*	`-namingservicehost` *string*	Use this property to specify the host on which the naming service is running. The default is `""`.
NamingServiceIORfile *string*	`-namingserviceiorfile` *string*	Use this property to specify the Naming Service object reference file. The default is `""`.
NamingServicePort *integer*	`-namingserviceport` *integer*	Use this property to specify the port on which the Naming Service is running. The default is `0`.

Table 6. Properties and command line options (continued)

Property name	Command line option	Description
NotificationCategories *string*	`-notification categories` *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* 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 *string*	`-notificationirpior` *string*	Use this property to specify the Notification IRP object. The default is `""`.
NotificationIRPIORFile *string*	`-notificationirpiorfile` *string*	Use this property to specify the path to the Notification IRP IOR file. The default is `""`.
NotificationIRPName *string*	`-notificationirpname` *string*	Use this property to specify the name of the Notification IRP object to which the probe sends active alarm subscription requests. The default is `""`.
ORBCharEncoding *string*	`-orbcharencoding` *string*	Use this property to specify the native character encoding set used by the Object Request Broker (ORB) for character data. The default is `ISO-8859-1`.
ORBDebug *string*	`-orbdebug` *string*	Use this property to specify whether the probe writes ORB messages to a debug log file. This property takes the following values: `false`: The probe does not write ORB messages to a debug file. `true`: The probe writes ORB messages to a debug file. The default is `false`.
ORBDebugFile *string*	`-orbdebugfile` *string*	Use this property to specify the location of the ORB debug file. The default is `$OMNIHOME/log/orb.debug`.

Table 6. Properties and command line options (continued)		
Property name	**Command line option**	**Description**
ORBInitialHost *string*	`-orbinitialhost` *string*	Use this property to specify the host name of the Naming Service server. The default is `""`.
ORBInitialPort *integer*	`-orbinitialport` *integer*	Use this property to specify the port number through which to connect to the Naming Service host. The default is `1570`.
ORBLocalHost *string*	`-orblocalhost` *string*	Use this property to specify the host name or IP address of the host where the application server or client application ORB is running. The default is `""`.
ORBLocalPort *integer*	`-orblocalport` *integer*	Use this property to specify the port number for the ORB to listen on. The default is `0`.
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.

Release3GPP *string*	`-release3gpp` *string*	Use this property to specify the version of 3GPP that the host is running. The possible values are:
		`V3.2`
		`V5.5.1`
		`V6.3`
		`V6.4`
		`V7.0`
		`V9.1`
		The default is `V9.1`.

SecurityProtocol *string*	`-securityprotocol` *string*	Use this property to specify the security protocol. This property takes the following values:
		`TLS`
		`TLSv1`
		`TLSv1.2`
		The default is `TLSv1`.

Table 6. Properties and command line options (continued)		
Property name	**Command line option**	**Description**
TimeTick *integer*	`-timetick` *integer*	Use this property to specify the time (in minutes) that device sessions are kept open. The default is `-1` (this instructs the probe to keep sessions open permanently). **Note :** If the generic **HeartBeatInterval** property is set to `0`, the **Retry** and **RetryInterval** properties will no affect. So if **TimeTick** is enabled, it should be used with the **Inactivity** property to shut down the probe following disconnection by the target system. If the **HeartBeatInterval** property is set to a value greater than `0`, and if the heartbeat interval is shorter than the time tick interval, time tick will not be triggered to terminate connections because the expiry time of the connections will be constantly renewed on each heartbeat.

Properties and command line options provided by the Java Probe Integration Library (probe-sdk-java) version 4.0

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 4.0.

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

Table 7. 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 7. 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 `60`.
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).
InitialResync *string*	`-initialresync` *string*	Use this property to specify whether the probe requests all active alarms from the host server 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.
MaxEventQueueSize *integer*	`-maxeventqueue size`*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 `10000`. **Note :** You can increase this number to increase the event throughput when a large number of events is generated.

Table 7. Properties and command line options (continued)		
Property name	**Command line option**	**Description**
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).
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 8. 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.
$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.

Table 8. Elements (continued)

Element name	Element description
$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.
$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.

Table 8. Elements (continued)	
Element name	**Element description**
`$AlarmRaisedTime`	This element specifies the time at which the event was raised.
`$AlarmClearedTime`	This element specifies the time at which the event was cleared.

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 9. Error messages		
Error	**Description**	**Action**
`Fail to retrieve java.class.path system property`	The Java classpath has not been set.	Set the Java classpath.
`Exception when trying to load jar to classpath.`	Unable to find JAR file for target system	Check that the JAR file for the target system is in the following directory: `$OMNIHOME/probes/java/corba` Check that the JAR file is readable.

Table 9. Error messages (continued)

Error	Description	Action
`Error connecting to EMS IRP`	Unable to connect to the intended IRP object.	Check that the IRP properties are set correctly in the properties file. If connecting through the Entry Point Integration Reference Point (EPIRP), check the values set for the following properties: • **EntryPointIORFile** • **AlarmIRPName** • **NotificationIRPName** If connecting through IOR files for the Alarm IPR and Notification IRP, check the values set for the following properties: • **AlarmIRPIORFile** • **NotificationIRPIORFile** Check that the server containing the IRP object is running and that the network is accessible.
`Failed to load class`	Unable to find JAR file for target system.	Check that the JAR file for the target system is in the following directory: `$OMNIHOME/probes/java/corba` Check that the JAR file is readable.
`Unsupported 3GPP release`	An unknown or unsupported 3GPP target version has been set using the **Release3GPP** property.	Set the **Release3GPP** property to a valid value in the properties file.
`Failed to retrieve 3GPP Release`	The **Release3GPP** property has not been set.	Set the **Release3GPP** property to a valid value in the properties file.

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 10. ProbeWatch messages		
ProbeWatch message	**Description**	**Triggers or causes**
`Running ...`	The probe is running normally.	The probe has just been started up.
`START SYNCHRONIZATION`	Resynchronization is in progress.	The probe started up, or the period specified by the **`ResyncInterval`** property has elapsed since the last resynchronization.
`END SYNCHRONIZATION`	The probe is ending the resynchronization process.	The probe has successfully received the active alarm list.
`Going Down ...`	Probe is shutting down.	The probe is shutting down after performing the shutdown routine.

Known issues

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

Multibyte characters not supported in some fields

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

- `AdditionalText`: The display of multibyte characters in this field is not currently supported in versions 3.2, 5.5.1, 6.3, 6.4, 7.0, 9.1 of the 3GPP interface.
- `ManagedObjectInstance`: The display of multibyte characters in this field is not currently supported in version 3.2 of the 3GPP interface.

Chapter 2. Migrating from existing probes

The Probe for Nokia-Siemens Switch/Radio/@vantage Commander (CORBA) 3GPP collects alarms from the following element management systems (EMS):

- Nokia-Siemens Switch Commander
- Nokia-Siemens Radio Commander
- Nokia-Siemens @vantage Commander

The Generic 3GPP Probe can also monitor the same systems. This chapter contains guidance on how to migrate from the Probe for Nokia-Siemens Switch/Radio/@vantage Commander (CORBA) 3GPP to the generic probe. The migration procedure has the following stages:

1. Install the generic probe.
2. Migrate the properties file.
3. Customize the rules file.
4. Run and test the generic probe.
5. 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 Generic 3GPP Probe in production once you are sure that it behaves in the same way as the probe it is replacing.

Comparison of probe features

All probes have some features in common, others are specific to the generic probe or to one of the existing, product-specific probes.

Common features

The following features are common to all the 3GPP interface probes:

Table 11. Features common to all 3GPP 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.
Data acquisition	Ability to receive alarms and notifications. Filtering for notifications and alarms. Peer-to-peer failover functinality. Support for Unicode and non-Unicode characters.

Features specific to the Generic 3GPP Probe

The Generic 3GPP Probe has the following additional features that are not present in one or more of the product-specific probes:

- “Command line interface” on page 30
- “HTTP/HTTPS command interface” on page 30

Command line interface

For sites that use Netcool/OMNIbus V7.3.1 and earlier, the generic probe has a command line interface (CLI) that enables you to connect to the probe over Telnet and manage the probe. Commands are available to acknowledge and clear alarms, perform a resynchronization with the NMS or EMS, obtain the size of the event queue, and shut down the probe. Availability of the CLI is controlled by the **CommandPort** and **CommandPortLimit** properties. "Command line interface" on page 8 has more information on the interface and the available commands.

HTTP/HTTPS command interface

For sites that use Netcool/OMNIbus 7.4 and later, the generic probe has a HTTP/HTTPS command interface. This enables you to send commands to the probe in JSON over a HTTP or HTTPS connection. Commands are available to acknowledge and clear alarms, and perform a resynchronization. "HTTP/HTTPS command interface" on page 9 has more information on the interface, how to configure it, and the format of the commands. There are also examples of each command.

Migration procedure

Use this procedure to replace the Probe for Nokia-Siemens Switch/Radio/@vantage Commander (CORBA) 3GPP with the generic probe.

Installing the Generic 3GPP Probe

Follow the advice in "Installing probes" on page 3 to download and install the generic probe in to a test environment.

Migrating properties

Determine the values required for the properties file of the generic probe. These properties are described in the following topics:

Use the properties file for the Probe for Nokia-Siemens Switch/Radio/@vantage Commander 3GPP to set the correct values for the generic probe.

Differing property names

The generic probe uses different names for some properties to those used in the Probe for Nokia-Siemens Switch/Radio/@vantage Commander 3GPP, as shown in the following table.

Table 12. Properties with different names in the generic probe

System-specific property	Generic property
Agentheartbeat	**HeatbeatInterval**
EntryPointIrpFile	**EntryPointIORFile**
NotificationIrpFile	**NotificationIRPIORFile**

Table 12. Properties with different names in the generic probe (continued)	
System-specific property	**Generic property**
NotificationIrpName	**NotificationIRPName**
ORBLocalHostName	**ORBLocalHost**
Resynch	**InitialResync** **ResyncInterval**
Retry	**RetryCount** **RetryInterval**
Timeout	**Inactivity**

Note : The **Release3GPP** property is found only in the generic probe. Set this property to the version of the 3GPP standard that the NMS or EMS implements.

Customizing the rules file

Edit the rules file for the generic probe to:

- Apply any vendor-specific enrichment or filtering that the generic rules file does not provide.
- Migrate custom rules from the system-specific rules file to the generic rules file.
- Apply changes to the @ClassID, @Manager, and lookup tables as required.

Note : The generic probe may not be able to parse certain attributes if the vendor does not follow the 3GPP standard or has implemented their own types that are not 3GPP compliant.

Attributes

There are some differences in the names or values of attributes between the system-specific probes and the generic probe. The following table indicates where there are differences, and shows the element that the 3GPP standard defines. Be sure to make the necessary changes if you copy over rules from the Probe for Nokia-Siemens Switch/Radio/@vantage Commander 3GPP rules file.

Table 13. Differences in rules file attributes		
3GPP element name	**Probe for Nokia-Siemens Switch/Radio/@vantage Commander 3GPP,**	**Generic 3GPP Probe**
ACK_STATE	NV_ACK_STATE	$AckState
ACK_SYSTEM_ID	NV_ACK_SYSTEM_ID	$AckSystemId
ACK_TIME	NV_ACK_TIME	$AckTime
ACK_USER_ID	NV_ACK_USER_ID	$AckUserId
ADDITIONAL_TEXT	NV_ADDITIONAL_TEXT	$AdditionalText
ALARM_CLEARED_TIME	Not available.	$AlarmClearedTime
ALARM_ID	NV-ALARM_ID	$AlarmId
ALARM_LIST_ALLIGNMENT_ REQUIREMENT	NV_ALARM_LIST_ALLIGNMENT_ REQUIREMENT	$AlarmListAllignment Requirement

Table 13. Differences in rules file attributes (continued)

3GPP element name	Probe for Nokia-Siemens Switch/Radio/@vantage Commander 3GPP,	Generic 3GPP Probe
ALARM_RAISED_TIME	Not available.	$AlarmRaisedTime
BACKUP_OBJECT	BACKUP_OBJECT	$BackupObject
BACKUP_UPSTATUS	NV_BACKUP_UPSTATUS	$BackupUpStatus
CLEAR_SYSTEM_ID	NV_CLEAR_SYSTEM_ID	$ClearSystemId
CLEAR_USER_ID	CLEAR_USER_ID	$ClearUserId
COMMENTS	NV_COMMENTS	$Comments
CORRELATED_NOTIFICATIONS	NV_CORRELATED_ NOTIFICATIONS	$CorrelatedNotifications
Not applicable.	DOMAIN_NAME	domain_name
Not applicable.	EVENT_NAME	EventName
Not applicable.	EVENT_TYPE	EventType
EVENT_TIME	NV_EVENT_TIME	$EventTime
MANAGED_OBJECT_CLASS	NV_MANAGED_OBJECT_CLASS	$ManagedObjectClass
MANAGED_OBJECT_INSTANCE	NV_MANAGED_OBJECT_INSTANC E	$ManagedObjectInstance
MONITORED_ATTRIBUTES	NV_MONITORED_ATTRIBUTES	$MonitoredAttributes
NOTIFICATION_ID	NV_NOTIFICATION_ID	$NotificationID
PERCEIVED_SEVERITY	NV_PERCEIVED_SEVERITY	$PerceivedSeverity
PROBABLE_CAUSE	NV_PROBABLE_CAUSE	$ProbableCause
PROPOSED_REPAIRACTIONS	NV_PROPOSED_REPAIR_ACTION S	$ProposedRepairActions
REASON	NV_REASON	$Reason
SECURITY_ALARM_DETECTOR	NV_SECURITY_ALARMDETECTOR	$SecurityAlarmDetector
SERVICE_PROVIDER	NV_SERVICE_PROVIDER	$ServiceProvider
SERVICE_USER	NV_SERVICE_USER	$ServiceUser
SPECIFIC_PROBLEM	NV_SPECIFIC_PROBLEM	$SpecificProblem

Table 13. Differences in rules file attributes (continued)		
3GPP element name	**Probe for Nokia-Siemens Switch/Radio/@vantage Commander 3GPP,**	**Generic 3GPP Probe**
STATE_CHANGE_DEFINITION	NV_STATE_CHANGE_ DEFINITION	$StateChangeDefinition
SYSTEM_DN	NV_SYSTEM_DN	$SystemDN
THRESHOLD_INFO	NV_THRESHOLD_INFO	$ThresholdInfo
TREND_INDICATION	NV_TREND_INDICATION	$TrendIndication
VENDOR_SPECIFIC_ALARM_TYP E	NV_VENDOR_SPECIFIC_ALARM_ TYPE	$VendorSpecificAlarmType

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/probes/nco_p_generic_3gpp -messagelog stdout -messagelevel debug -
   initialresync true
   ```

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

   ```
   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 "Device connections through the CORBA interface" on page 6 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-UNK-000-000: Probewatch: START SYNCHRONIZATION
   Debug: D-JPR-000-000: Filter value is : sadsadsa
   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 "Properties and command line options provided by the Java Probe Integration Library (probe-sdk-java) version 4.0" on page 20 for information on synchronization.

4. Check that the probe correctly parses alarms with the Event Processor. Check for any unsupported types for event parsing. For example:

   ```
   Debug: D-UNK-000-000: [Event Processor] Reason:    SOME REASON
   Debug: D-UNK-000-000: [Event Processor] EventName:    x2
   ```

```
Debug: D-UNK-000-000: [Event Processor] NotificationID:    997169651
Debug: D-UNK-000-000: [Event Processor] AckState:    1
Debug: D-UNK-000-000: [Event Processor] AlarmId:    676512950
Debug: D-UNK-000-000: [Event Processor] SpecificProblem:
                                    supervision inhibited by operator
Debug: D-UNK-000-000: [Event Processor] BackupUpStatus:    false
Debug: D-UNK-000-000: [Event Processor] AdditionalText_LineCount:    1
Debug: D-UNK-000-000: [Event Processor] AckSystemId:    ack_system_id
Debug: D-UNK-000-000: [Event Processor] TrendIndication:    MoreSevere
Debug: D-UNK-000-000: [Event Processor] FullDN:    (NW, UTRAN)
                                (BTSEquipment.userLabel, BTSM2_413_005)
Debug: D-UNK-000-000:
         [Event Processor] ManagedObjectClass:    managed_object_class
```

5. Check the log file for errors that occur from parsing unsupported types of event. For example:

```
Cannot parse event attribute 'X.733:CorrelatedNotifications' with type [19]
and content description: Sequence
```

   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.

Part Number:

(1P) P/N:

SC27-6561-02