# IBM® Tivoli® Netcool/OMNIbus Probe for Huawei U2000 (CORBA) 4.0

*Reference Guide*
*August 9, 2018*

SC27-6541-04

**Notice**

Before using this information and the product it supports, read the information in Appendix A, “Notices and Trademarks,” on page 45.

**Edition notice**

This edition (SC27-6541-04) applies to version 4.0 of Probe for Huawei U2000 (CORBA) and to all subsequent releases and notifications until otherwise indicated in new editions.

This edition replaces SC27-6541-03.

# 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 (CORBA) documentation is provided in softcopy format only. To obtain the most recent version, visit the IBM® Tivoli® Knowledge Center:

http://www-01.ibm.com/support/knowledgecenter/SSSHTQ/omnibus/common/kc_welcome-444.html?lang=en

*Table 1. Document modification history*

| Document version | Publication date | Comments |
|---|---|---|
| SC27-6541-00 | June 12, 2014 | First IBM publication. |
| SC27-6541-01 | November 7, 2014 | "Summary" on page 1 updated. |
| SC27-6541-02 | March 10, 2016 | Guide updated for version 3.0 of the probe.<br>Support extended to the following target systems:<br>• Huawei iManager U2000 V100R006C02<br>"Summary" on page 1 updated.<br>Descriptions for the **`acknowledgeAlarm`** and **`unacknowledgeAlarm`** commands to "Commands supported by the probe over Telnet" on page 22.<br>"Error messages" on page 41 updated.<br>"ProbeWatch messages" on page 42 updated.<br>Version 3 of the probe addresses the following enhancement requests:<br>• **RFE 76265**: Support for acknowledge and unacknowledge operations added.<br>• **RFE 71705**: Support for using the host name as a lookup in the Interoperable Object Reference added.<br>• **RFE 36131**: ProbeWatch extension for Huawei U2000 Corba Probe. Additional probewatch message is sent after the last re-synchronized alarm received.<br>Version 3 of the probe also contains a fix for the following APAR:<br>• **IV65119**: The probe fails to start when the ORBLocalPort is set, but runs correctly when ORBLocalPort is not set. |

*Table 1. Document modification history (continued)*

| Document version | Publication date | Comments |
|---|---|---|
| SC27-6541-03 | July 20, 2017 | References to `nco_g_crypt` removed from the guide.<br>This version of the guide has been updated address the following APAR:<br>• **IV96122**: `nco_g_crypt` not able to encrypt for `.props` |
| SC27-6541-04 | August 9, 2018 | Guide updated for version 4.0 of the probe.<br>"Summary" on page 1 updated.<br>"Notification caching" on page 18 added.<br>Description for the following new properties added to "Properties and command line options" on page 27:<br>• **`EnableNotificationCaching`**<br>• **`NotificationCacheInterval`**<br>• **`NotificationCacheSize`**<br>`$isResynch` token introduced to the probe rules. This token takes the value `true` for resynchronization alarms and `false` for notifications.<br>This version of the guide has been updated for the following enhancements:<br>• **RFE 49616**: Enhancement to the probe parser to process the `CommunicationState_T` attribute.<br>• **RFE 51573**: Enhancement to process synchronization events first before processing notification events. |

## 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. Probe for Huawei U2000 (CORBA)

The Huawei U2000 is a unified network management system that provides element management and network management functions for telecommunications networks.

The Probe for Huawei U2000 (CORBA) acquires data from the Huawei U2000 element management system (EMS) using 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.

**Note :** This probe is not supported on Windows.

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 U2000 V100R006C00<br>Huawei iManager U2000 V100R006C02<br>Huawei iManager U2000 V100R009C00 |
| Probe executable name | `nco_p_huawei_u2000_corba` |
| Package version | 4.0 |
| Probe supported on | For details of supported operating systems, see the following Release Notice on the IBM Software Support website:<br>http://www-01.ibm.com/support/docview.wss?uid=swg21450078 |

*Table 3. Summary (continued)*

| | |
|---|---|
| Properties file | `$OMNIHOME/probes/`*`arch`*`/huawei_u2000_corba.props` |
| Rules file | `$OMNIHOME/probes/`*`arch`*`/huawei_u2000_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 |
| 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*.

# Migrating from Huawei T2000 to Huawei U2000

The Probe for Huawei U2000 replaces the Probe for Huawei T2000. This topic describes how to migrate to the Probe for Huawei U2000.

## Functionality supported by the two probes

The Probe for Huawei U2000 supports most of the functionality supported by the Probe for Huawei T2000, and also supports some additional functionality. The following table shows the functionality supported by the two probes.

*Table 4. Supported features*

| Functionality | Probe for Huawei T2000 | Probe for Huawei U2000 |
|---|---|---|
| SSL connectivity | Yes | Yes |
| Event synchronization | Yes | Yes |
| Server failover | Yes | Yes |
| IOR file | Yes | Yes |
| Naming service host/port | Yes | Yes |
| Naming service IOR file | Yes | Yes |
| Resynchronization | Yes | Yes |
| Resynchronization with interval | No | Yes |
| Resynchronization filter with severity | Yes | Yes |
| Resynchronization filter with probable cause | No | Yes |
| Resynchronization with batch | Yes | Yes |
| Notification | Yes | Yes |
| Persistence notification | Yes | No |
| Heartbeat check status | Yes | Yes |
| Inactivity and timeout | Yes | Yes |
| Reconnect and backoff | Yes | Yes |
| Multibyte character support | Yes | Yes |
| Interactive command port | Yes | Yes |
| Stream capture | No | Yes |
| ORB debug logging | No | Yes |

## Migrating properties

The Probe for Huwaei U2000 uses different names for some properties to those used by the earlier Probe for Huawei T2000. The following table shows the names of the properties used by the Probe for Huawei T2000 and their equivalents used by the Probe for Huawei U2000. For details of configuring the various properties listed, refer to the data acquisition topics that follow.

*Table 5. Properties names used by the old and the new probe*

| Functionality | Probe for Huawei T2000 properties | Probe for Huawei U2000 properties |
|---|---|---|
| Authentication | **Username**<br>**Password** | **Username**<br>**Password** |
| SSL connectivity | **ClientCertificate**<br>**ClientCertificatePassword**<br>**EnabledProtocols**<br>**EnableSSL**<br>**SecurityProtocol**<br>**TrustStore**<br>**TrustStorePassword** | **EnableSSL**<br>**KeyStore**<br>**KeyStorePassword**<br>**SecurityProtocol** |
| Event synchronization | **EventSynchronization** | **EventSynchronization** |
| Server failover | **EnableFailover**<br>**SecondaryIORFile**<br>**SecondaryNamingContextIORFile**<br>**SecondaryORBInitialHost**<br>**SecondaryORBInitialPort**<br>**ServerSwitchingTime** | **EnableFailover**<br>**SecondaryIORFile**<br>**SecondaryNamingServiceIORFile**<br>**SecondaryNamingServiceHost**<br>**SecondaryNamingServicePort**<br>**RetryInterval** |
| IOR file | **IORFile** | **IORFile** |
| Naming service host/port | **ORBInitialHost**<br>**ORBInitialPort**<br>**NamingContextPath** | **NamingServiceHost**<br>**NamingServicePort**<br>**NamingContextPath** |
| Naming service IOR file | **NamingContextIORFile** | **NamingServiceIORFile** |
| Resynchronization | **Resynch** | **InitialResync** |
| Resynchronization with interval | Functionality not supported by this probe. | **ResyncInterval** |

*Table 5. Properties names used by the old and the new probe (continued)*

| Functionality | Probe for Huawei T2000 properties | Probe for Huawei U2000 properties |
|---|---|---|
| Resynchronization filter with severity | **ExcludeSeverityCleared**<br>**ExcludeSeverityCritical**<br>**ExcludeSeverity Indeterminate**<br>**ExcludeSeverityMajor**<br>**ExcludeSeverityMinor**<br>**ExcludeSeverityWarning** | **ResyncSeverityFilter**<br>**Note :** There is no direct correspondence between the properties that filter alarms by severity. From the Probe for Huawei T2000 properties file, determine the severities that you want to filter and then construct the appropriate value for the **ResyncSeverityFilter** property. |
| Resynchronization filter with probable cause | Functionality not supported by this probe. | **ResyncProbableCauseFilter** |
| Resynchronization with batch | **ResynchBatchSize** | **ResyncBatchSize** |
| Persistence notification | **PersistenceFile**<br>**PersistentNotification** | Functionality not supported by this probe. |
| Heartbeat check status | **AgentHeartbeat** | **HeartbeatInterval** |
| Inactivity and timeout | **Timeout** | **Inactivity** |
| Reconnect and backoff | **Retry** | **RetryCount**<br>**RetryInterval** |
| Notification stream capture | Functionality not supported by this probe. | **StreamCapture**<br>**StreamCaptureFilePath** |
| ORB debug logging | Functionality not supported by this probe. | **ORBDebug**<br>**ORBDebugFile** |

## Configuring the probe

After installing the probe you need to make various configuration settings to suit your environment.

The following table outlines how to use the probe's properties to configure the product's features. Configuration of some features is mandatory for all installations. For those features set the properties to the correct values or verify that their default values are suitable for your environment. Further configuration is optional depending on which features of the probe you want to use.

*Table 6. Configuring the probe*

| Feature | Properties | See |
|---|---|---|
| **Mandatory features:** | | |

*Table 6. Configuring the probe (continued)*

| Feature | Properties | See |
|---|---|---|
| **CORBA connection method**<br>The method that the probe obtains the reference to the object needed to connect to the CORBA interface. | **IORFile**<br>**NamingServiceHost**<br>**NamingServicePort**<br>**NamingServiceIORFile**<br>**NamingContextPath** | "Connecting to the CORBA interface" on page 10 |
| **Authentication**<br>Credentials for authenticating with the EMS. | **Password**<br>**Username** | "Authentication" on page 16 |
| **Optional features:** | | |
| **Resynchronization policy**<br>Specifies whether the probe resynchronizes with the EMS. | **InitialResync**<br>**ResyncInterval**<br>**ResyncBatchSize**<br>**ResyncProbableCauseFilter**<br>**ResyncSeverityFilter** | "Alarm retrieval and synchronization" on page 17 |
| **Reconnection policy**<br>Specifies whether the probe attempts to reconnect to the EMS following a communications failure. | **RetryCount**<br>**RetryInterval** | "Reconnection and probe backoff strategy" on page 19 |
| **Inactivity policy**<br>Specifies whether the probe disconnects from the EMS following a period of inactivity. | **Inactivity** | "Inactivity" on page 19 |
| **Heartbeat policy**<br>Specifies whether the probe periodically checks that the connection to the EMS endpoint is still operational. | **HeartbeatInterval** | "Heartbeat" on page 19 |
| **Support for Unicode and non-Unicode characters**<br>Enables the probe to process alarms that contain characters encoded in UTF-8, such as Asian languages. | **EncodingStandard**<br>**ORBCharEncoding**<br>**ORBWCharDefault** | "Support for Unicode and non-Unicode characters" on page 20 |

*Table 6. Configuring the probe (continued)*

| Feature | Properties | See |
|---|---|---|
| **Peer-to-peer failover pair**<br>Allows you to set up two probes to act as a failover pair to improve availability. If the master probe should stop working, the slave probes takes over until the master is available once more. | **`MessageFile`**<br>**`Mode`**<br>**`PeerHost`**<br>**`PeerPort`**<br>**`PidFile`**<br>**`PropsFile`**<br>**`RulesFile`** | "Peer-to-peer failover functionality" on page 21 |
| **Command line interface (Telnet)**<br>Defines the port allocated to receive CLI commands sent over Telnet, and defines the maximum number of concurrent Telnet connections. | **`CommandPort`**<br>**`CommandPortLimit`** | "Commands supported by the probe over Telnet" on page 22 |
| **HTTP/HTTPS command interface**<br>Enables the HTTP/HTTPS command interface and defines the port that it uses. | **`NHttpd.EnableHTTP`**<br>**`NHttpd.ListeningPort`**<br>**`NHttpd.ExpireTimeout`** | "Commands supported by the probe over HTTP/HTTPs" on page 24 |

## 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 firewall settings

The Probe for Huawei U2000 (CORBA) can be used in conjunction with a firewall.

If a firewall exists between the probe and the target system, you must configure the following firewall ports to enable data flow:

- The Naming Service port number.

  The port for this service is on the target system. You must also specify this port on the probe side of the firewall, using the **`ORBInitialPort`** property. The default value of the **`ORBInitialPort`** property is `1570`.
- The notification service port number.

  The port for this service is on the target system. This port number is usually random but can be set to a fixed value by the administrator of the target system. No firewall configuration is required on the probe side.
- The CORBA Agent (ORB at target system or inter-ORB bridge) port number.

  The port number is specified both on the target system and on the probe side of the firewall. On the probe side, use the **`OrbLocalPort`** property to specify this value, which must be greater than `0`. The port number on the target system can be obtained from either the Naming Service or the Interoperable Object Reference file (specified by the **`IORFile`** property).

When using the probe over Secure Sockets Layer (SSL) connections, you must configure the same ports listed above.

## Making the probe NIST compliant

The National Institute of Standards and Technology (NIST) defines standards for measuring equipment and procedures, quality control benchmarks for industrial processes, and experimental control samples. Products sold within US Federal markets must comply with SP800-131a.

You can configure the probe to support the NIST SP800-131a security standard. SP800-131a requires longer key lengths and stronger cryptography than other standards, for example, FIPS 140-2. SP800-131a requires Transport Layer Security (TLS) V1.2. To make the probe NIST compliant, there are two considerations:

1. The vendor's EMS must be able to support the signature algorithm and key length that is NIST compliant. The key provided must be generated using the signature algorithm SHA2 (or above) with the RSA key length greater than or equal to 2048. This you must then convert into PKCS12 format before importing into the keystore using the IBM KeyMan utility. For details of the conversion and importing process, see "SSL-based connectivity" on page 9.
2. The security protocol must be set to TLSv1.2 (or above). To specify that the probe uses protocol TLSv1.2, set the **`SecurityProtocol`** property accordingly.

**Note :** You can access the full SP800-131a standard at the following address:

http://csrc.nist.gov/publications/nistpubs/800-131A/sp800-131A.pdf

## SSL-based connectivity

The Probe for Huawei U2000 (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

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. 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.

# Running the probe

Probes can be run in a variety of ways. The way you chose depends on a number of factors, including your operating system, your environment, and the any high availability considerations that you may have.

For details about how to run the probe, visit the following page on the IBM Tivoli Knowledge Center:

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

# 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 IBM Tivoli Netcool/OMNIbus Probe for Huawei U2000 (CORBA) gathers events from the EMS using a CORBA interface.

Data acquisition is described in the following topics:

- "Connecting to the CORBA interface" on page 10
- "Authentication" on page 16
- "Alarm retrieval and synchronization" on page 17
- "Notification caching" on page 18
- "Server failover" on page 18
- "Retrieving alarms" on page 18
- "Reconnection and probe backoff strategy" on page 19
- "Inactivity" on page 19
- "Heartbeat" on page 19
- "Data stream capture" on page 19
- "Support for Unicode and non-Unicode characters" on page 20
- "Peer-to-peer failover functionality" on page 21

## Connecting to the CORBA interface

The probe connects to the target system through a CORBA interface.

To complete the connection, the probe needs a reference to the EmsSessionFactory_I object. The following topics contain:

- A summary of the methods that the probe can use to obtain the reference to the EmsSessionFactory_I object.
- Instructions on how to configure the probe for each of those methods.
- Advice on how to use the messages in the log file to help confirm that you have configured the probe correctly or help you solve configuration problems.

### Methods for obtaining a reference to the EmsSessionFactory_I object

The probe can obtain the object reference in one of two ways:

- Using an IOR file
- Using a Naming Service

### Using an IOR file

When using Interoperable Object Reference (IOR) files, the probe obtains the reference to the EmsSessionFactory_I CORBA object from the IOR file specified in the **IORFile** property.

### Using a Naming Service

As an alternative to an IOR file, the probe can use a Naming Service to obtain the reference to the EmsSessionFactory_I object. There are two ways that the probe can locate the Naming Service:

- By using the host name and port number of the Naming Service specified in the **NamingServiceHost** and **NamingServicePort** properties.
- By using the IOR file specified in the **NamingServiceIORFile** property.

The Naming Service then uses the value specified in the **NamingContextPath** property to obtain the reference to the EmsSessionFactory_I object.

### Completing the connection sequence

Once the probe has obtained the reference to the EmsSessionFactory_I object, it logs in to the target system. It then creates an EMS session and queries the Subscriber and EMS Manager objects. The probe uses the Subscriber object to subscribe to real-time event notifications and the EMS Manager object to perform resynchronization operations.

## Configuring the probe

Use the following procedure to configure the probe:

1. Decide on the method you want to use to obtain the reference to the EmsSessionFactory_I object.
2. Define values for the properties listed in the section for your chosen method.

### IOR file

Set the **IORFile** property to the path for the Interoperable Object Reference (IOR) file used to connect to the target through CORBA. For example:

```
IORFile = "/opt/var/emssession.ior"
```

### Locating the Naming Service using a specified host and port

Set the following properties:

- **NamingServiceHost**: Set this property to the name of the host server that provides the Naming Service.
- **NamingServicePort**: Set this property to the port on the host server to use to connect to the Naming Service.
- **NamingContextPath**: Set this property to the full path of the EmsSessionFactory_I interface on the target system.

  The following article in the Service Management Connect (SMC) technical community on IBM developerWorks shows how to construct the value of this property:

  https://www.ibm.com/developerworks/community/wikis/home?lang=en#/wiki/Tivoli%20Netcool%20OMNIbus/page/How%20to%20configure%20naming%20context%20path%20for%20TMF%20standard%20Corba%20probe

For example:

```
NamingServiceHost = "nshost1"
NamingServicePort = "8054"
```

```
NamingContextPath = "TMF_MTNM.Class/TejasNetworks.Vendor/TejasNetworks\\
/NORTH-CDG.EmsInstance/3\\.5.Version/TejasNetworks\\/NORTH-CDG.EmsSessionFactory_I"
```

### Locating the Naming Service using an IOR file

Set the following properties:

- **`NamingServiceIORFile`** : Set this property to the path for the IOR file for the Naming Service.
- **`NamingContextpath`**: Set this property to the path of the `EmsSessionFactory` interface.

  The following article in the Service Management Connect (SMC) technical community on IBM developerWorks shows how to construct the value of this property:

  https://www.ibm.com/developerworks/community/wikis/home?lang=en#/wiki/Tivoli%20Netcool%20OMNIbus/page/How%20to%20configure%20naming%20context%20path%20for%20TMF%20standard%20Corba%20probe

For example:

```
NamingServiceIORFile = "/opt/var/ns.ior"
NamingContextPath = "TMF_MTNM.class/test/EmsSessionfactory_I"
```

## Messages in the log file

Use the probe's log file to confirm that you configured the probe correctly or to help you solve configuration errors. The following sections contain the messages that appear for various connection situations.

### Successful connection

This example shows the messages that occur in the log file on successfully connecting to the target system through a Naming Service running on a specified host and port:

```
Debug: D-JPR-000-000: Attempting to get reference for interface object
Debug: D-JPR-000-000: Attempting to get object reference via ORBInitialHost and
ORBInitialPort settings
Debug: D-JPR-000-000: Attempting to connect to Naming Service via host and
port settings
Debug: D-JPR-000-000: Sucessfully connected
Debug: D-JPR-000-000: Attempting to resolve Naming Context to object reference
Debug: D-JPR-000-000: Narrowing reference to NamingContext
Debug: D-JPR-000-000: Successfully narrowed reference to Naming Context
Debug: D-JPR-000-000: Resolving Object  reference :
      TMF_MTNM.Class/TejasNetworks.Vendor/TejasNetworks\/NORTH-CDG.EmsInstance/3
\.5.Version/TejasNetworks\/NORTH-CDG.EmsSessionFactory_I
Debug: D-JPR-000-000: Resolved Object reference
Debug: D-JPR-000-000: Successfully found object reference
Debug: D-JPR-000-000: Narrowing object reference to interface object
```

### No properties configured

The following example shows the messages that appear in the log file when none of the CORBA properties are configured:

```
Warning: W-JPR-000-000: NamingContextPath is empty, please ensure you have set
this property if you want to connect to CORBA via Naming service
Error: E-JPR-000-000: IOR Object is null. Please check your probe settings.
Error: E-JPR-000-000: Failed to get IOR Object : IOR Object is null.
Error: E-JPR-000-000: Failed to connect: com.ibm.tivoli.netcool.omnibus.probe.
ProbeException: IOR Object is null.
```

## IORFile property refers to an incorrect or invalid IOR file

The following example shows the messages that appear in the log file when the **IORFile** property refers to an incorrect or invalid IOR file:

```
Information: I-JPR-000-000: Read IOR file /home/netcool/sim/dist/var/ems.ior
Debug: D-JPR-000-000: com.ibm.tivoli.netcool.omnibus.probe.bidi.CommandHandler.
registerTarget ENTERING
Debug: D-JPR-000-000: com.ibm.tivoli.netcool.omnibus.probe.bidi.CommandHandler.
registerTarget EXITING
Information: I-JPR-000-000: Converting string IOR to object reference :
IOR:000000000000002B49444C3A6F6D672E6F72672F436F734E616D696E672F4E616D696E67436F6
E746578744578743A312E3000000000000001000000000000000780001020000000000A3132372E302E31
2E310026250000001F5374616E646172644E532F4E616D6553657276657722D504F412F5F726F6F740
0000000200000000000000000800000004A4143000000000001000000240000000000501000100000002
000100010001000F00010109000000020501000100010100
Information: I-JPR-000-000: Retrieveing EMS Session via IOR object...
Error: E-JPR-000-000: Failed to get IOR Object :
Error: E-JPR-000-000: Failed to connect: org.omg.CORBA.BAD_PARAM:   vmcid: 0x0
minor code: 0  completed: No
```

## Cannot connect when using the IORFile property

The following example shows the messages that can appear in the log file in these circumstances:

- Invalid host, port, or host and port specified in he IOR file
- Firewall issues
- The target system is not online

```
Information: I-JPR-000-000: Read IOR file /home/netcool/sim/dist/var/
emssessionfactory.ior
Information: I-JPR-000-000: Converting string IOR to object reference :
IOR:000000000000003F49444C3A6D746E6D2E746D666F72756D2E6F72672F656D7353657373696F
6E466163746F72792F456D7353657373696F6E466163746F72795F493A312E300000000000010000
0000000007800010200000000A3132372E302E312E310084B80000001F37383136373533383838
2F04221F100E034A10161006304638141 41B484C1B000000000200000000000000008000000004A41
4300000000010000002400000000050100010000000200010001000 1000F00010109000000020501
000100010100
Information: I-JPR-000-000: Retrieveing EMS Session via IOR object...
Error: E-JPR-000-000: Failed to get interface version information:
org.omg.CORBA.TRANSIENT: initial and forwarded IOR inaccessible  vmcid: IBM
minor code: E07  completed: No
Error: E-JPR-000-000: Failed to connect: org.omg.CORBA.TRANSIENT:
initial and forwarded IOR inaccessible  vmcid: IBM  minor code: E07
completed: No
```

## Not all properties set when connecting through a Naming Service host and port

The following example shows the messages that can appear in the log file when connecting through a Naming Service using a specified host and port. In this instance one of the **NamingServiceHost**, **NamingServicePort** and **NamingContextPath** properties has no value:

```
Warning: W-JPR-000-000: NamingContextPath is empty, please ensure you have
set this property if you want to connect to CORBA via Naming service
Error: E-JPR-000-000: IOR Object is null. Please check your probe settings.
Error: E-JPR-000-000: Failed to get IOR Object : IOR Object is null.
Error: E-JPR-000-000: Failed to connect: com.ibm.tivoli.netcool.omnibus.probe.
ProbeException: IOR Object is null.
Debug: D-JPR-000-000: com.ibm.tivoli.netcool.omnibus.probe.ProbeException:
com.ibm.tivoli.netcool.omnibus.probe.ProbeException: IOR Object is null.
```

## Properties have incorrect values when connecting through a Naming Service host and port

The following example shows the messages that can appear in the log file when connecting through a Naming Service using a specified host and port. In this instance, one or more of the

**NamingServiceHost**, **NamingServicePort** and **NamingContextPath** properties has an incorrect value, or the host, port, or path is inaccessible:

```
Error: E-JPR-000-000: Failed to resolve initial references to the NamingService :
NameService:org.omg.CORBA.COMM_FAILURE: purge_calls:2004 Reason: CONN_ABORT (1),
State: ABORT (5)  vmcid: IBM  minor code: 306 completed: Maybe
Error: E-JPR-000-000: Failed to get IOR Object :
org.omg.CORBA.ORBPackage.InvalidName: NameService:org.omg.CORBA.COMM_FAILURE:
purge_calls:2004 Reason: CONN_ABORT (1),
State: ABORT (5)  vmcid: IBM  minor code: 306 completed: Maybe
Error: E-JPR-000-000: Failed to connect:
com.ibm.tivoli.netcool.omnibus.probe.ProbeException:
org.omg.CORBA.ORBPackage.InvalidName: NameService:org.omg.CORBA.COMM_FAILURE:
purge_calls:2004 Reason: CONN_ABORT (1), State: ABORT (5)  vmcid: IBM
minor code: 306 completed: Maybe
```

## Firewall configuration preventing connection to a Naming Server host or port

The following example shows the messages that can appear in the log file when configuration problems with a firewall prevent connection to the host or server of a Naming Service:

```
Debug: D-JPR-000-000: Resolving initial references to NamingService
Error: E-JPR-000-000: Failed to resolve initial references to the NamingService :
NameService:org.omg.CORBA.TRANSIENT: java.net.ConnectException:
Unable to connect:host=127.0.0.1,port=9765  vmcid: IBM  minor code: E02
completed: No
Error: E-JPR-000-000: Failed to get IOR Object :
org.omg.CORBA.ORBPackage.InvalidName: NameService:org.omg.CORBA.TRANSIENT:
java.net.ConnectException: Unable to connect:host=127.0.0.1,port=9765  vmcid: IBM
minor code: E02  completed: No
Error: E-JPR-000-000: Failed to connect:
com.ibm.tivoli.netcool.omnibus.probe.ProbeException:
org.omg.CORBA.ORBPackage.InvalidName: NameService:org.omg.CORBA.TRANSIENT:
java.net.ConnectException: Unable to connect:host=127.0.0.1,port=9765  vmcid: IBM
minor code: E02  completed: No
```

## Incorrect NamingContextPath or the Naming Server is offline

The following example shows the messages that can appear in the log file in these circumstances:

- An incorrect value for the **NamingContextPath** property means the ORB is unable to narrow the configured context path on the target system.
- The host server for the Naming Service is offline.

```
Error: E-JPR-000-000: Failed to get the System reference from the naming
service! :
IDL:omg.org/CosNaming/NamingContext/NotFound:1.0
Error: E-JPR-000-000: Failed to resolved to Naming Context :
org.omg.CosNaming.NamingContextPackage.NotFound:
IDL:omg.org/CosNaming/NamingContext/NotFound:1.0
Error: E-JPR-000-000: Failed to get IOR Object :
com.ibm.tivoli.netcool.omnibus.probe.ProbeException:
org.omg.CosNaming.NamingContextPackage.NotFound:
IDL:omg.org/CosNaming/NamingContext/NotFound:1.0
Error: E-JPR-000-000: Failed to connect:
com.ibm.tivoli.netcool.omnibus.probe.ProbeException:
com.ibm.tivoli.netcool.omnibus.probe.ProbeException:
org.omg.CosNaming.NamingContextPackage.NotFound:
IDL:omg.org/CosNaming/NamingContext/NotFound:1.0
```

## IORFile or NamingServiceIORFile specifies an incorrect path

The following example shows the messages that can appear in the log file when the path specified by the **IORFile** or **NamingServiceIORFile** is incorrect and the probe cannot find the IOR file:

```
Error: E-JPR-000-000: Failed to get object from Naming Service IOR  file:
Failed to find file /home/netcool/sim/dist/var/em.ior:
java.io.FileNotFoundException: /home/netcool/sim/dist/var/em.ior
```

### NamingServiceIORFile specifies an incorrect IOR file or the IOR is incorrect

The following example shows the messages that appear in the log file when the value of the **NamingServiceIORFile** property refers to an incorrect IOR file or to a file that specifies and incorrect IOR:

```
Error: E-JPR-000-000: Failed to connect to the NamingService :
Error: E-JPR-000-000: Failed to resolve to the naming context:
org.omg.CORBA.BAD_PARAM:   vmcid: 0x0  minor code: 0  completed: No
Error: E-JPR-000-000: Failed to get IOR Object :
com.ibm.tivoli.netcool.omnibus.probe.ProbeException:
org.omg.CORBA.BAD_PARAM:   vmcid: 0x0  minor code: 0  completed: No
Error: E-JPR-000-000: Failed to connect:
com.ibm.tivoli.netcool.omnibus.probe.ProbeException:
com.ibm.tivoli.netcool.omnibus.probe.ProbeException:
org.omg.CORBA.BAD_PARAM:
```

## Diagnosing the naming service connection

The CORBA probe framework is supplied with two utilities that allow you to diagnose the naming service connection. These can help you to troubleshoot any connection related issues that the probe may have.

### dumpns

This utility allows you display the naming context of a session. The dumpns script takes as arguments the naming service host and port and returns the naming context string. This is a wrapper script for `org.jacorb.naming.ContextLister`.

dumpns takes the following format:

dumpns *nshost nsport*

Where *nshost* is the host and *nsport* is the port of the naming service whose naming context string you want to return.

### Example Usage

```
> $OMNIHOME/probes/java/corba/jacorb-3.3/bin/dumpns 127.0.0.1 1570
/opt/ibm/tivoli/nco740/omnibus/probes/java/corba/jacorb-3.3/bin/jaco org.jacorb.naming.ContextLister
 -url corbaloc:iiop:127.0.0.1:1570/NameService
May 22, 2014 9:48:23 AM org.jacorb.orb.ORBSingleton <init>
INFO: created ORBSingleton
May 22, 2014 9:48:23 AM org.jacorb.orb.portableInterceptor.InterceptorManager <init>
INFO: InterceptorManager started with 0 Server Interceptors, 0 Client Interceptors and 1 IOR Interceptors
May 22, 2014 9:48:23 AM org.jacorb.orb.giop.ClientConnectionManager getConnection
INFO: ClientConnectionManager: created new ClientGIOPConnection to 127.0.0.1:1570 (bf7f82d6)
May 22, 2014 9:48:23 AM org.jacorb.orb.iiop.ClientIIOPConnection connect
INFO: Connected to 127.0.0.1:1570 from local port 59928
May 22, 2014 9:48:23 AM org.jacorb.orb.giop.ClientConnectionManager getConnection
INFO: ClientConnectionManager: found ClientGIOPConnection to 127.0.0.1:1570 (bf7f82d6)
   TMF_MTNM.Class/
May 22, 2014 9:48:23 AM org.jacorb.orb.giop.ClientConnectionManager getConnection
INFO: ClientConnectionManager: found ClientGIOPConnection to 127.0.0.1:1570 (bf7f82d6)
May 22, 2014 9:48:23 AM org.jacorb.orb.giop.ClientConnectionManager getConnection
INFO: ClientConnectionManager: found ClientGIOPConnection to 127.0.0.1:1570 (bf7f82d6)
        HUAWEI.Vendor/
May 22, 2014 9:48:23 AM org.jacorb.orb.giop.ClientConnectionManager getConnection
INFO: ClientConnectionManager: found ClientGIOPConnection to 127.0.0.1:1570 (bf7f82d6)
May 22, 2014 9:48:23 AM org.jacorb.orb.giop.ClientConnectionManager getConnection
INFO: ClientConnectionManager: found ClientGIOPConnection to 127.0.0.1:1570 (bf7f82d6)
                Huawei\/U2000.EmsInstance/
May 22, 2014 9:48:23 AM org.jacorb.orb.giop.ClientConnectionManager getConnection
INFO: ClientConnectionManager: found ClientGIOPConnection to 127.0.0.1:1570 (bf7f82d6)
May 22, 2014 9:48:23 AM org.jacorb.orb.giop.ClientConnectionManager getConnection
INFO: ClientConnectionManager: found ClientGIOPConnection to 127.0.0.1:1570 (bf7f82d6)
                        2\.0.Version/
May 22, 2014 9:48:23 AM org.jacorb.orb.giop.ClientConnectionManager getConnection
INFO: ClientConnectionManager: found ClientGIOPConnection to 127.0.0.1:1570 (bf7f82d6)
May 22, 2014 9:48:23 AM org.jacorb.orb.giop.ClientConnectionManager getConnection
INFO: ClientConnectionManager: found ClientGIOPConnection to 127.0.0.1:1570 (bf7f82d6)
                                Huawei\/U2000.EmsSessionFactory_I
   NotificationService
```

```
May 22, 2014 9:48:23 AM org.jacorb.orb.ORB shutdown
INFO: prepare ORB for shutdown...
May 22, 2014 9:48:23 AM org.jacorb.orb.ORB shutdown
INFO: ORB going down...
May 22, 2014 9:48:23 AM org.jacorb.orb.iiop.ClientIIOPConnection close
INFO: Client-side TCP transport to 127.0.0.1:1570 closed.
May 22, 2014 9:48:23 AM org.jacorb.orb.ORB shutdown
INFO: ORB shutdown complete

Naming Context: TMF_MTNM.Class/HUAWEI.Vendor/Huawei\/U2000.EmsInstance/2\.0.Version/Huawei\/
U2000.EmsSessionFactory_I
```

### dior

This utility allows you to decode an interoperable object reference (IOR) in string form into a more readable representation. The `dior` script is provided by JacORB for `org.jacorb.orb.util.PrintIOR`. It prints the IOR components in detail.

`dior` takes the following format:

`dior -i` *`ior_str`*

Where *`ior_str`* is the path of the IOR file whose details you want to print.

### Example Usage

```
> $OMNIHOME/probes/java/corba/jacorb-3.3/bin/dior -i `cat ns.ior` 2>/dev/null
------IOR components-----
TypeId  :       IDL:omg.org/CosNaming/NamingContextExt:1.0
TAG_INTERNET_IOP Profiles:
        Profile Id:             0
        IIOP Version:           1.2
        Host:                   127.0.0.1
        Port:                   1570
        Object key (URL):       StandardNS/NameServer-POA/_root
        Object key (hex):       0x53 74 61 6E 64 61 72 64 4E 53 2F 4E 61 6D 65 53
65 72 76 65 72 2D 50 4F 41 2F 5F 72 6F 6F 74
        -- Found 2 Tagged Components--
        #0: TAG_ORB_TYPE
                Type: 1245790976 (JacORB)
        #1: TAG_CODE_SETS
                ForChar native code set Id: ISO8859_1
                Char Conversion Code Sets: ISO8859_15, UTF8
                ForWChar native code set Id: UTF16
                WChar Conversion Code Sets: UTF8, UCS2
```

## Authentication

Once the probe has obtained a reference to the `EmsSessionFactory_I` object, it logs in to the target using the values stored in the **`Username`** and **`Password`** properties. The value of the **`Password`** property can be plain text or an AES encrypted password. To encrypt a password, use the **nco_keygen** utility to create a key file and then use the **nco_aes_crypt** utility to encrypt the password using the key file.

Detailed instructions on how to encrypt a property value, such as **`Password`** are in the *IBM Tivoli Netcool/OMNIbus Installation and Deployment Guide*. The following example shows how to encrypt the password:

1. Use **nco_keygen** to create a key file; for example:

   `$NCHOME/omnibus/bin/nco_keygen -o $NCHOME/omnibus/probes/key_file`

2. Set the value of the probe's **`ConfigKeyFile`** property to the file path of the key file; for example:

   `ConfigKeyFile: "$NCHOME/omnibus/probes/key_file"`

3. Set the value of the probe's **`ConfigCryptoAlg`** property to AES:

   `ConfigCryptoAlg: "AES"`

4. Use **nco_aes_crypt** to encrypt the password; for example:

   `$NCHOME/omnibus/bin/nco_aes_crypt -c AES -k key_file password`

5. Set the value of the probe's **Password** property to the encrypted string generated by **nco_aes_crypt**; for example:

   ```
   Password: "@44:U/ccVZ0K+ftc7gZTV33Yx2fODe5v46RZzEbvqpE=@"
   ```

## Alarm retrieval and synchronization

On startup, the probe can retrieve active alarms from the EMS and do so regularly if required. The probe uses the CORBA push model to receive new alarms as the EMS generates them.

### Startup and initial synchronization

At startup, the probe retrieves a list of all active alarms from the EMS if the **InitialResync** property is set to `true`. When the property is set to `false`, the probe does not receive the existing alarms.

### Alarm retrieval

Once the probe has received any existing alarms, it connects to the Subscriber object and uses the CORBA notification push model to receive new alarms from the EMS. The probe receives those alarms as they are generated at the EMS.

The probe parses each alarm it receives and forwards it to the ObjectServer.

### Resynchronization

The probe can resynchronize with the EMS periodically. The frequency of any resynchronization is determined by the value of the **ResyncInterval** property. When the property has a value of `0`, which is the default value, the probe never resynchronizes. Any other value of **ResyncInterval** defines the interval, in seconds, between successive resynchronization operations. For each operation the probe receives a list of all active alarms in the same way as it does at startup. The probe then resumes waiting for new alarms from the EMS. When resynchronizing, the probe receives alarms in batches when the **ResyncBatchSize** property has a positive value (the default value is `100`). The minimum batch size is `1`.

During normal operation, the probe requests a resynchronization at one of the following trigger points:

- Initial resynchronization: This is performed when the probe starts, and is controlled by **InitialResync** property. This uses the resynchronization filters configured in the probe properties file.
- Interval resynchronization: This is performed while the probe is running, and is controlled by **ResyncInterval** property. This also uses the resynchronization filters and is similar to the initial resynchronization.
- `resync` command: This is performed from the command port without a filter.
- `resyncFilter` command: This is performed from the command port with a filter passed to the command.

When the resynchronization operation is in progress, attempts to launch the command line resynchronization will be aborted with the response:

```
I-UNK-104-002: {"response":["Resync in progress. Abort command line
resync."],"status":"200"}
```

**Note :** The probe is supplied with a script (`EventSynch_U2000.sh`) which removes events that have an occurrence date and time that is prior to that of the resynchronization ProbeWatch message. This script is triggered for the initial resynchronization, the interval resynchronization, and the `resync` command. The script is not triggered for the `resyncFilter` command.

### Resynchronization filters

You can apply filters during a resynchronization operation to limit the number of alarms returned from the EMS. The probe provides two properties that enable you to define filters:

- **ResyncProbableCauseFilter**
- **ResyncSeverityFilter**

You can use either filter individually or both filters together.

The filters define values for alarms to exclude from a resynchronization operation when they contain a particular value. For example, if you set **ResyncSeverityFilter** to the value `PS_MINOR`, all alarms with that severity setting are excluded from the resynchronization operation.

## Notification caching

Notification caching occurs during resynchronization of all modes (initial, periodic, and command line) if the **EnableNotificationCaching** property is set to `true`.

This functionality withholds notifications from event parsing until one of the following conditions is met:

1. Alarm resynchronization completes, or
2. Any notification thresholds that have been applied are met.

The threshold properties are **NotificationCacheInterval** and **NotificationCacheSize**. These properties specify the duration that the probe holds notifications and the cache size that the probe uses to store notifications, respectively. To apply a threshold, set the associated property to a non-zero positive value. If a threshold is not applied, there are no constraints imposed on that aspect.

## Retrieving alarms

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

## Server failover

This feature enables failover between the probe and the primary and secondary EMS servers. During failover, the probe will continue to switch between the primary and secondary server until a connection is made to one of them.

Server failover is configured using the **EnableFailover**, **RetryCount**, and **RetryInterval** properties.

**Note :** To enable the server failover function, the following conditions must be met:

1. The **EnableFailover** property must be set to `true` to enable the values specified in the following properties:
   - **SecondaryIORFile**
   - **SecondaryNamingContextIORFile**
   - **SecondaryNamingServiceHost**
   - **SecondaryNamingServicePort**
2. The **RetryCount** property must be set to a value greater then `0`.

   When the **RetryInterval** property is set to `0`, the probe will continue to retry the connection based on backoff strategy interval, up to the number of attempts specified by the **RetryCount** property.
3. The value of the **RetryInterval** property must not exceed 4096 seconds.

   This is the default value of the backoff strategy time that, if exceeded, will cause the probe to disconnect from the CORBA interface.

## Reconnection and probe backoff strategy

Use the **RetryCount** and **RetryInterval** properties to specify how the probe reacts if the connection to the target system is lost or cannot be established.

Use the **RetryCount** property to specify whether the probe attempts to reconnect to the target system. Setting the property to 0, the default value, means that the probe does not try to reconnect and simply shuts down. Any other, positive value specifies the number of times the probe tries to reconnect before shutting down.

Use the **RetryInterval** property to specify the number of seconds between each attempt to reconnect to the target system. Setting the property to 0 means that the probe uses an exponentially increasing interval between connection attempts. First the probe waits 1 second, then 2 seconds, then 4 seconds, and so on up to a maximum of 4095 seconds. If this limit, or the number of connection attempts is reached, the probe shuts down.

## 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.

## Heartbeat

The probe can disconnect from the target system if the connection between them becomes unavailable.

You can use the **HeartbeatInterval** property to specify whether the probe periodically checks that the connection to the target system is available and how often it performs that check. The probe shuts down if it detects that the connection to the target system is unavailable.

When the **HeartbeatInterval** property has a value of 0 the probe does not check the availability of the connection. Any other positive value defines the number of seconds between each check of the connection's availability.

**Note :** Once the probe shuts down it may restart again, depending on the value set for the **RetryCount** property. If the value set for **RetryCount** is 0, the probe does not restart. For any other positive value the probe follows the reconnection policy. See "Reconnection and probe backoff strategy" on page 19 for more information.

To check the connection to the target system, the probe sends a ping command (using the standard function EmsSession_I_ping) and waits for a response from the target system.

The probe also disconnects from the target system if it receives an endSession request from the EMS. This may occur if the target system restarts or is shut down.

## Data stream capture

The probe can capture the stream of binary data from the EMS and store it in a file. The data can be used for debugging purposes, to develop new features for the probe, or to pass onto other management systems that require the same data.

To capture the data stream in log files, use the following procedure:

1. Set the value of the **StreamCapture** property to 1.
2. Set the value of the **StreamCaptureFilePath** property to the full path of a directory to hold the files of data.

   **Notes :**

   - Specify the full path of the directory. For example:

`/opt/tivoli/netcool/omnibus/var`

- You cannot include variables such as `$OMNIHOME` in the directory path.
- The directory must exist. The probe does not create the directory if it does not exist.

3. If the probe is running, restart the probe.

The probe now writes stream data to the specified directory. The probe creates two types of file: one contains resynchronization data and the other contains notification data. The names for these files have the following format:

- Resynchronization data file:

  `resync-`*`timestamp`*`-`*`n`*`.evtraw`
- Notification data file:

  `notif-`*`timestamp`*`-`*`n`*`.evtraw`

In both file names *timestamp* is the time of day when the file was created, in milliseconds and *n* is a sequence number for the file. The number increases by one for each file that is created.

Example:

`notif-137111893172-0.evtraw`

The probe creates a separate file for each event it receives from the endpoint.

**Note :** Capturing the data stream to a log file generates a lot of data, consuming a lot of disk space and other system resources. So use this feature with caution. As soon as you no longer require the capture of data, set the value of the **`StreamCapture`** property to `0` and restart the probe.

## Support for Unicode and non-Unicode characters

The probe can process multibyte characters and so can display both Unicode and non-Unicode characters.

Use the following procedure to set up the probe to process multibyte characters:

1. Ensure that the EMS is configured to send data in UTF-8 format.
2. Set the appropriate locale on the system that runs the probe by changing the values of the **`LANG`** and **`LC_ALL`** environment variables. For example, to set the locale to simplified Chinese, use the following commands:

```
export LANG=zh_CN.utf8
export LC_ALL=zh_CN.utf8
```

3. Set the following properties of the probe:

| **Property** | **Value** |
|---|---|
| **`EncodingStandard`** | `UTF-8` |
| **`ORBCharEncoding`** | UTF8 |
| **`ORBWChardefault`** | UTF16 |

4. Configure the ObjectServer to enable the insertion of data that uses UTF-8 encoding. The *IBM Tivoli Netcool/OMNIbus Administration Guide* shows how to create, configure, and run an ObjectServer in UTF-8 mode.
5. Run the probe or restart it, if it is already running.

## 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"
```

# Command line interface

The probe is supplied with a command line interface (CLI) that allows you to manage the probe while it is running.

## Managing the probe over a telnet connection

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.

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. The default port number is 7777.
- **`CommandPortLimit`**: Set this to the maximum number of CLI connections that can be open concurrently.

**Note :** If you are running the probe on Netcool/OMNIbus V7.4.0 (or later) you can also manage the probe over an HTTP/HTTPS connection.

## Commands supported by the probe over Telnet

The following table describes the commands that the probe supports over `Telnet`.

*Table 7. Commands supported over Telnet*

| Command | Description |
|---|---|
| **acknowledgeAlarm** *alarmIds* | Use this command to acknowledge one or more alarms, where *alarmIds* is a comma-separated list of identifiers of the alarms that you want to acknowledge. |
| **exit/quit** | Use this command to close the connection. |
| **help** | Use this command to display online help about the CLI. |
| **resync** | Use this command to perform a resynchronization using the values specified by the **ResyncSeverityFilter** and **ResyncProbableCauseFilter** properties. |
| **resyncFilter** *filter* | Use this command to perform a resynchronization using a custom filter.<br>Custom filters take the following format:<br>**resyncFilter** sev=*severity1*;*severity2*; pbCause=*pbCause1*;*pbCause2*;<br>To perform a full resynchronization, use the following command:<br>**resyncFilter** sev= pbCause=<br>**Note :** This command does not accept spaces in the `sev` and `pbCause` arguments. If you want to use a filter that contains spaces, you must use the **ResyncSeverityFilter** and **ResyncProbableCauseFilter** properties. |
| **stop** | Use this command to shut down the probe. |
| **unacknowledgeAlarm** *alarmIds* | Use this command to unacknowledge one or more alarms, where *alarmIds* is a comma-separated list of identifiers of the alarms that you want to unacknowledge. |
| **version** | Use this command to print 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 Netcool/OMNIbus Event List.

## Managing the probe over an HTTP/HTTPS connection

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 elapsed time (in seconds) that and 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_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:

  **acknowledgeAlarm**
  **help**
  **resync**
  **resyncFilter**
  **stop**
  **unacknowledgeAlarm**

- *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.

## Commands supported by the probe over HTTP/HTTPs

The following sections define the structure of the 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://test1.example.com and a HTTP listening port of 6789.

### acknowledgeAlarm

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

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

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

Where:

- *alarmIds* is a comma-separated list of identifiers of the alarms that you want to acknowledge.

The following example acknowledges the alarms with IDs 111 and 222:

```
$OMNIHOME/bin/nco_http -uri http://test1.example.com:6789/probes/
huawei_u2000_corba -datatype application/JSON -method POST -data
'{"command":"acknowledgeAlarm", "params":[{"alarmIds":"111,222"}]}'
```

### 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://test1.example.com:6789/probes/
huawei_u2000_corba -datatype application/JSON -method POST -data
'{"command":"help", "params":[]}'
```

The response from the probe includes the following message:

```
Information: I-UNK-104-002: {"response":"Available commands: ackAlarm(alarmId
String,emsId String,managedElementId String,username String),
unackAlarm(alarmId String,emsId String,managedElementId String,username
String), resync(), resyncFilter(excludeSeverity String,excludePbCause String),
stop() ","status":"200"}
```

### resync

Use the **resync** command to perform a resynchronization with the endpoint using the value specified by the **ResyncSeverityFilter** and **ResyncProbableCauseFilter** properties.

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

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

The following example resynchronizes the probe:

```
$OMNIHOME/bin/nco_http -uri http://test1.example.com:6789/probes/
huawei_u2000_corba -datatype application/JSON -method POST -data
'{"command":"resync", "params":[]}'
```

### resyncFilter

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

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

```
-data '{"command":"resyncFilter","params":[{"excludeSeverity":"sev=severities",
"excludePbCause":"pbCause=probable-causes"}]}'
```

Where:

- *severities* is a list of severities to exclude when the probe resynchronizes with the CORBA interface. Separate each entry in the list with a semicolon.
- *probable-causes* is a list of probable causes to exclude when the probe resynchronizes with the CORBA interface. Separate each entry in the list with a semicolon.

The following example resynchronizes the probe an excludes alarms with a severity of `PS_CLEARED` or `PS_WARNING`:

```
$OMNIHOME/bin/nco_http -uri http://test1.example.com:6789/probes/
huawei_u2000_corba -datatype application/JSON -method POST -data
'{"command":"resyncFilter", "params":
[{"excludeSeverity":"sev=PS_CLEARED;PS_WARNING",
"excludePbCause":"pbCause="}]}'
```

### stop

Use the **`stop`** command to shut down the probe.

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

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

The following example stops the probe:

```
$OMNIHOME/bin/nco_http -uri http://test1.example.com:6789/probes/
huawei_u2000_corba -datatype application/JSON -method POST -data
'{"command":"stop", "params":[]}'
```

### unacknowledgeAlarm

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

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

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

Where:

- *alarmIds* is a comma-separated list of identifiers of the alarms that you want to unacknowledge.

The following example unacknowledges the alarms with IDs 111 and 222:

```
$OMNIHOME/bin/nco_http -uri http://test1.example.com:6789/probes/
huawei_u2000_corba -datatype application/JSON -method POST -data
'{"command":"unacknowledgeAlarm", "params":[{"alarmIds":"111,222"}]}'
```

## 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 the 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
```

## 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 information about common properties and command line options, see the *IBM Tivoli Netcool/OMNIbus Probe and Gateway Guide*.

*Table 8. Properties and command line options*

| Property name | Command line option | Description |
|---|---|---|
| **`EnableFailover`** *string* | `-disenablefailover` (This is equivalent to **`EnableFailover`** with a value of `false`.)<br>`-enablefailover` (This is equivalent to **`EnableFailover`** with a value of `true`.) | Use this property to specify whether the server failover function between the probe and the primary and secondary EMS servers is enabled or disabled. This property takes the following values:<br>`false`: The server failover function between the probe and the EMS servers is disabled.<br>`true`: The server failover function between the probe and the EMS servers is enabled.<br>The default is `false`. |
| **`EnableNotificationCaching`** *string* | `-disablenotificationcaching` (This is equivalent to **`EnableNotificationCaching`** with a value of `false`.)<br>`-enablenotificationcaching` (This is equivalent to **`EnableNotificationCaching`** with a value of `true`.) | Use this property to control whether notifications are withheld from event parsing during alarm resynchronization. This property takes the following values:<br>`false`: Notifications are not withheld from event parsing during alarm resynchronization.<br>`true`: Notifications are withheld from event parsing during alarm resynchronization.<br>The default is `false`.<br>**Note :** If you set the property to `true`, you can set notification caching thresholds using the **`NotificationCacheInterval`** and **`NotificationCacheSize`** properties. For details, see "Notification caching" on page 18. |

*Table 8. Properties and command line options (continued)*

| **Property name** | **Command line option** | **Description** |
|---|---|---|
| **EnableSSL** *string* | `-disenablessl` (This is equivalent to **EnableSSL** with a value of `false`.)<br>`-enablessl` (This is equivalent to **EnableSSL** with a value of `true`.) | Use this property to specify whether SSL connectivity between the probe and the EMS server is enabled or disabled. This property takes the following values:<br>`false`: SSL connectivity between the probe and the EMS server is disabled.<br>`true`: SSL connectivity between the probe and the EMS server is enabled.<br>The default is `false`. |
| **EncodingStandard** *string* | `-encodingstandard` *string* | Use this property to specify the character encoding standard that the probe uses. Possible values for this property are:<br>`ISO-8859-1`: This sets the encoding standard to Latin Alphabet 1.<br>`UTF-8`: This sets the encoding standard to UTF-8.<br>The default is: `ISO-8859-1` |
| **EventSynchronization** *string* | `-noeventresynch` (This is equivalent to **EventSynchronization** with a value of `false`.)<br>`-eventresynch` (This is equivalent to **EventSynchronization** with a value of `true`.) | Use this property to synchronize the ObjectServer with the EMS. This removes the alarms from the ObjectServer that were cleared in the EMS while the probe was not running. This property takes the following values:<br>`false`: The probe does not synchronize with the EMS on startup.<br>`true`: The probe synchronizes with the EMS on startup.<br>The default is `false`. |
| **IORFile** *string* | `-iorfile` *string* | Use this property to specify the path of the Interoperable Object Reference (IOR) file used to connect to the target through the CORBA interface. If you do not provide a value for this property, use the **NamingContextPath** with the **NamingServiceHost** and **NamingServicePort** properties or the **NamingServiceIORFile** property to define the Naming Service to use instead.<br>The default is `""`. |

*Table 8. Properties and command line options (continued)*

| Property name | Command line option | Description |
|---|---|---|
| **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.<br>The default is "". |
| **KeyStorePassword** *string* | -keystorepassword *string* | Use this property to specify the password required to access the certificate specified by the **Keystore** property.<br>The default is "". |
| **NamingContextPath** *string* | -nspath *string* | Use this property to specify the location of the object in the Naming Service. If using a Naming Service to connect to the CORBA interface, always set this property.<br>The default is "". |
| **NamingServiceHost** *string* | -nshost *string* | Use this property to specify the name of the host that runs the Naming Service. If you do not use the **IORFile** property define the location of the Naming Service using this property, together with the **NamingServicePort** and **NamingContextPath**properties, or the **NamingServiceIORFile** and **NamingContextPath** properties.<br>The default is: "". |
| **NamingServiceIORFile** *string* | -nsiorfile *string* | Use this property to specify the location of the IOR file that contains the root context of the Naming Service. If you do not provide a value for the **IORFile** property, use this property and the **NamingContextPath** property or the **NamingServiceHost**, **NamingServicePort**, and **NamingContextPath** properties to define the Naming Service to use to obtain the reference to the EmsFactorySession_I object.<br>The default is: "". |

*Table 8. Properties and command line options (continued)*

| **Property name** | **Command line option** | **Description** |
|---|---|---|
| **`NamingServicePort`** *string* | `-nsport` *string* | Use this property to specify the port on the host defined by **`NamingServiceHost`** through which to connect to the Naming Service. If you do not use the **`IORFile`** property, use this property, together with the **`NamingServiceHost`** property and **`NamingContextPath`** property, or the **`NamingServiceIORFile`** and **`NamingContextPath`** properties.<br>The default is: `0`. |
| **`NotificationCacheInterval`** *integer* | `-notificationcacheinterval` *integer* | Use this property to specify how long (in seconds) the probe withholds notifications from event parsing during alarm resynchronization. If you set this property to zero, this threshold will not be enabled.<br>The default is: `0`.<br>**Note :** This property works in conjunction with the **`EnableNotificationCaching`** property. |
| **`NotificationCacheSize`** *integer* | `-notificationcacheszie` *integer* | Use this property specify the number of notifications that the probe withholds from event parsing during alarm resynchronization. If you set this property to zero, this threshold will not be enabled.<br>The default is: `0`.<br>**Note :** This property works in conjunction with the **`EnableNotificationCaching`** property. |
| **`ORBCharEncoding`** *string* | `-orbcharencoding` *string* | Use this property to specify the native character encoding set that the Object Request Broker (ORB) uses for character data. Possible values for this property are:<br>`ISO8859_1`<br>`UTF8`<br>The default is: `ISO8859_1`. |

*Table 8. Properties and command line options (continued)*

| Property name | Command line option | Description |
|---|---|---|
| **ORBDebug** *string* | -noorbdebug (This is equivalent to **ORBDebug** with a value of false.)<br>-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:<br>false: The probe does not write ORB debug messages to a log file.<br>true: The probe writes ORB debug messages to the log file specified by the **ORBDebugFile** property.<br>The default is false. |
| **ORBDebugFile** *string* | -orbdebugfile *string* | Use this property to specify the location of the file to which the probe writes ORB debug messages.<br>The default is "". |
| **ORBLocalHost** *string* | -orblocalhost *string* | Use this property to specify the local host used by the server-side ORB to place the server's host name or IP address into the IOR of a remote object.<br>The default is: "". |
| **ORBLocalPort** *integer* | -orblocalport *integer* | Use this property to specify the local port that the ORB listens on for connections from the probe.<br>The default is: 0 (the ORB selects a port at random). |
| **ORBWCharDefault** *string* | -orbchardefault *string* | Use this property to specify the wide character (wchar) set that the IBM ORB uses when communicating with other ORBs that do not publish a wchar set. Possible values for this property are:<br>UCS2<br>UTF16<br>The default is: UTF16. |
| **Password** *string* | -password *string* | Use this property to specify the password of the account to use when logging in to the target system. The password can be in plain text or encrypted using the AES algorithm. Always define a value for this property and the **Username** property.<br>The default is: "". |

*Table 8. Properties and command line options (continued)*

| Property name | Command line option | Description |
|---|---|---|
| **`PersistenceIORFile`** *string* | `-persistenceiorfile` *string* | Use this property to specify the location of the IOR file for the persistence notification service.<br>The default is: `""`. |
| **`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 of this property is `1`.<br>The default is: `100`. |
| **`ResyncProbableCause Filter`** *string* | `-resyncprobablecause filter` *string* | Use this property to specify a list of probable causes to exclude when the probe resynchronizes with the CORBA interface. Separate each entry in the list with a semicolon. For example:<br>`pbCause1;pbCause2;pbCause3`<br>The default is: `""`. |
| **`ResyncSeverityFilter`** *string* | `-resyncseverityfilter` *string* | Use this property to specify a list of severities that the probe excludes when resynchronizing with the CORBA interface. Separate each entry in the list with a semicolon. The severity values you can include are:<br>`PS_INDETERMINATE`<br>`PS_CRITICAL`<br>`PS_MAJOR`<br>`PS_MINOR`<br>`PS_WARNING`<br>`PS_CLEARED`<br>The default is: `""`. |
| **`SecondaryIORFile`** *string* | `-seciorfile` *string* | Use this property to specify the location of the IOR file of the secondary EMS server.<br>The default is `""`. |
| **`SecondaryNamingContext Path`** *string* | `-secnspath` *string* | Use this property to specify the location of the object of the secondary EMS within the Naming Service.<br>The default is `""`. |

*Table 8. Properties and command line options (continued)*

| Property name | Command line option | Description |
|---|---|---|
| **SecondaryNamingServiceHost** *string* | -secnshost *string* | Use this property to specify the Naming Service host name of the secondary EMS server.<br>The default is "". |
| **SecondaryNamingServiceIORFile** *string* | -secnsiorfile *string* | Use this property to specify the location of the interface object of the secondary EMS within the Naming Service.<br>The default is: "". |
| **SecondaryNamingServicePort** *string* | -secnsport *string* | Use this property to specify the Naming Service port number of the secondary EMS server.<br>The default is 0 . |
| **SecurityProtocol** *string* | -securityprotocol *string* | Use this property to specify the security protocol. This property takes the following values:<br>TLS<br>TLSv1<br>TLSv1.2<br>The default is TLSv1.<br>**Note :** For this probe to be compliant with NIST, set this property to TLSv1.2. For further details about NIST compliance, see "Making the probe NIST compliant" on page 8. |
| **StreamCapture** *integer* | -streamcapture *integer* | Use this property to specify whether the stream capture feature is enabled. The values this property can have are:<br>1: The probe uses the stream capture feature.<br>0: The probe does not use the stream capture feature.<br>The default is: 0.<br>**Note :** If you set the value of this property to 1, define a value for the **StreamCaptureFilePath** property as well. |

*Table 8. Properties and command line options (continued)*

| Property name | Command line option | Description |
|---|---|---|
| **StreamCaptureFilePath** *string* | -streamcapturefilepath *string* | Use this property to specify the directory where the probe stores the input data stream.<br>The default is: "".<br>See "Data stream capture" on page 19 for more information on how to use this property. |
| **Username** *string* | -username *string* | Use this property to specify the account to use when logging in to the target system. Always define a value for this property and the **Password** property.<br>The default is: "". |

## Properties and command line options provided by the Java Probe Integration Library (probe-sdk-java) version 9.0

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 9.0.

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

*Table 9. 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.<br>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.<br>The default is 10. |
| **DataBackupFile** *string* | -databackupfile *string* | Use this property to specify the path to the file that stores data between probe sessions.<br>The default is "".<br>**Note :** Specify the path relative to $OMNIHOME/var. |

*Table 9. Properties and command line options (continued)*

| Property name | Command line option | Description |
|---|---|---|
| **DisconnectionTimeout** *integer* | -disconnectiontimeout *integer* | Use this property to specify the maximum time, in seconds, for probe disconnection before shutting down the probe forcefully.<br>The default is 15. |
| **HeartbeatInterval** *integer* | -heartbeatinterval *integer* | Use this property to specify the frequency (in seconds) with which the probe checks the status of the host server.<br>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.<br>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 inactivity timeout is reached.<br>SHUTDOWN: Sends a ProbeWatch message to notify user and shuts down the probe.<br>CONTINUE: Sends a ProbeWatch message to notify user and do not shut down the probe.<br>The default is SHUTDOWN. |
| **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:<br>false: The probe does not request resynchronization on startup.<br>true: The probe requests resynchronization on startup.<br>For most probes, the default value for this property is false.<br>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 9. Properties and command line options (continued)*

| **Property name** | **Command line option** | **Description** |
|---|---|---|
| **MaxEventQueueSize** *integer* | -maxeventqueue sizeinteger | Use this property to specify the maximum number of events that can be queued between the non native process and the ObjectServer.<br>The default is 0.<br>**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.<br>For most probes, the default value for this property is 0 (which instructs the probe to not make successive resynchronization requests).<br>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.<br>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.<br>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). |

## 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 tables describe the elements that the Probe for Huawei U2000 generates. Not all the elements described are generated for each event; the elements that the probe generates depends upon the event type. The data type and, where applicable, enumeration values for these elements are defined in the TMF814 standard.

*Table 10. Elements specific to events*

| Element name | Element description |
|---|---|
| `$acknowledgeIndicaton` | This element indicates the acknowledgement status of the alarm. The possible values are:<br>• `AI_EVENT_ACKNOWLEDGED`<br>• `AI_EVENT_UNACKNOWLEDGED` |
| `$attributeList` | A list of attrbiutes and their values associated with the event. |
| `$DomainName` | This element identifies the TMF specification that defines the received event. For example:<br>`tmf_mtnm` |
| `$edgePointRelated` | This element indicates whether the event applies to a termination point that is an edge point. The element has the following possible values:<br>• `true`<br>• `false` |
| `$emsTime` | This element indicates the time at which the alarm was reported by the EMS. |
| `$EventName` | This element indicates the name of the event. |
| `$EventType` | This element indicates the type of the event. |
| `$groupName` | This element indicates the name of the group for this event. |
| `$isClearable` | This element indicates whether the alarm can be cleared. The possible values are:<br>`true`<br>`false` |
| `$layerRate` | This element indicates the layer to which the alarm applies. |
| `$nativeEMSName` | This element contains the name of the object reporting the alarm, as displayed in the EMS user interface.<br>**Note :** Object names must not contain brackets. If brackets are passed to the `$nativeEMSName` element as part of an object name, the node field in the event list is not filled. |
| `$neTime` | This elements indicates the time at which the error occurred in the network element. |

*Table 10. Elements specific to events (continued)*

| Element name | Element description |
|---|---|
| `$notificationID` | This element contains the unique identifier of the alarm. This is derived from the serial number of the alarm as used by the EMS. |
| `$objectName` | This element contains the name of the object reporting the alarm. |
| `$objectType` | This element contains the type of network object that the alert relates to. |
| `$perceivedSeverity` | This element contains the severity of the alert as perceived by the EMS. |

*Table 11. Elements specific to protection switch events*

| Element name | Element description |
|---|---|
| `$protectedE` | This identifies the protected equipment when the switch is made. |
| `$protectedTP` | This element identifies the protected termination point when the switch occurred. |
| `$ProtectionType` | This element identifies the type of the protection switch |
| `$switchAwayFromE` | This element identifies the source equipment where the switch is being made. |
| `$switchAwayFromTP` | This element identifies the source termination point where the switch is being made |
| `$switchReason` | This element indicates why the switch occurred. |
| `$switchToE` | This element identifies the destination equipment where the switch is being made. |
| `$switchToTP` | This element identifies the destination termination point where the switch is being made. |

*Table 12. Elements specific to alarms*

| Element name | Element description |
|---|---|
| `$additionalInfo` | This element contains additional information about the alarm. |
| `$additionalText` | This element contains a brief description of the problem being reported by the alarm. |
| `$affectedTPList` | This element identifies list of termination points affected by the problem being reported. |

*Table 12. Elements specific to alarms (continued)*

| Element name | Element description |
|---|---|
| $affectedPTPs | This element displays the list of termination points affected by the problem being reported. |
| $nativeProbableCause | This element indicates the probable cause as given in the EMS user interface. |
| $probableCause | This element contains the probable cause of the alarm. |
| $probableCauseQualifier | This element contains the qualifier used to classify the alarm type. |
| $rcaIndication | This element indicates whether an alarm is a root alarm. This alarm has the following possible values:<br>True (The alarm is a root alarm)<br>False (The alarm is a common alarm) |
| $serviceAffecting | This element indicates whether the alarm has affected the service. |
| $X.733::AdditionalInformation | This element indicates the ITU-T X733 additional information of the event. This consists of a list of up to five elements:<br>• $X733::AdditonalInfo_AlarmNO<br>• $X733::AdditionalInfo_DEVICE_LABEL<br>• $X733::AdditionalInfo_DeviceIP<br>• $X733::AdditionalInfo_Label<br>• $X733::AdditionalInfo_UserLabel |
| $X.733::BackUpObject | This element identifies the object that provides back-up services for the object that is the subject of an event. |
| $X.733::BackedUpStatus | This element indicates whether the object that is the subject of an event has been backed-up. It has the following possible values:<br>BACKED_UP<br>NOT_BACKED_UP<br>When the attribute has the value BACKED_UP, the value of the $X.733::BackUpObject identifies the back-up object. |

*Table 12. Elements specific to alarms (continued)*

| Element name | Element description |
|---|---|
| `$X.733::CorrelatedNotifications` | This element contains a list of root alarms that cause correlative alarms. This alarm has the following possible values:<br>• `soSource` (This field is always empty)<br>• `notifIDs` (For correlative alarms this field displays the serial number of the root alarm. For example, if alarm A causes alarm B, and alarm B causes alarm C this field will display alarm C along with the serial numbers of alarm A and alarm B.) |
| `$X.733::EventType` | This element contains the alarms classified into the following six basic types according to the ITU-T X.733:<br>• `communicationsAlarm`<br>• `qualityofServiceAlarm`<br>• `equipmentAlarm`<br>• `processingErrorAlarm`<br>• `securityAlarm`<br>• `environmentalAlarm` |
| `$X.733::MonitoredAttributes` | This element displays identifies one or more attributes of the managed object and their corresponding values at the time of the event. |
| `$X.733::ProposedRepairActions` | This element contains one or more proposed repair actions suggested by the EMS when it knows the probable cause and can suggest solutions to the event. |
| `$X.733::SpecificProblems` | This element identifies further refinements of the probable cause of an event. |
| `$X.733::TrendIndication` | This element compares the severity of the current alarm with that of all outstanding alarms raised for the object and indicates whether the severity has increased, decreased, or stayed the same. |

*Table 13. Elements specific to threshold crossing alerts*

| Element name | Element description |
|---|---|
| `$granularity` | This element contains the details of the threshold that has been crossed. |
| `$pmLocation` | This element indicates the `pmLocation` where the threshold has been crossed |
| `$pmParameterName` | This element contains the `pmParameter` that has crossed the threshold. |

*Table 13. Elements specific to threshold crossing alerts (continued)*

| Element name | Element description |
|---|---|
| `$thresholdType` | This element indicates whether a threshold was set for the log report. |
| `$unit` | This element contains the faulty program unit. |
| `$value` | This element contains the threshold value. |

## 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 error messages, see the *IBM Tivoli Netcool/OMNIbus Probe and Gateway Guide*.

*Table 14. Error messages*

| Error | Description | Action |
|---|---|---|
| `Alarm Error in acknowledge.` | The probe could not acknowledge an alarm specified in the **`ackAlarm`** command to the Telnet CLI or the HTTP/HTTPS command interface. | Check that the you have specified the parameters for the command correctly, in particular the alarm identifier. |
| `Alarm Error in unAcknowledge.` | The probe could not unacknowledge an alarm specified in the **`unackAlarm`** command to the Telnet CLI or the HTTP/HTTPS command interface. | Check that the you have specified the parameters for the command correctly, in particular the alarm identifier. |
| `Error in filtered resynchronization` | An error occurred while processing the **`resyncFilter`** command to the Telnet CLI or the HTTP/HTTPS command interface. | Check that you have specified the command correctly, in particular the attributes of the filter. |
| `Error in resynchronization` | An error occurred while processing the **`resync`** command to the Telnet CLI or the HTTP/HTTPS command interface. | Check that you have specified the command correctly. Also check that the values of the **`ResyncProbableCause`** and **`ResynchSeverityFilter`** are correctly specified. |
| `Cannot parse attribute` *`attribute-name`* `with type [`*`type`*`]` | The probe cannot parse the attribute named in the error message. | Check that the attribute is a supported type and is compliant with the TMF814 standard. Refer to the elements that the probe supports. You can also contact IBM Software Support for assistance. |

*Table 14. Error messages (continued)*

| Error | Description | Action |
|---|---|---|
| `Failed to close file` *ior_file* | The probe failed to close the specified IOR file. | Check that the IOR file is in the correct place, defined by the **`NamingServiceIORFile`** property or the **`IORFile`** property. In addition, check that the directory that holds the file is not set to read only. |
| `Failed to convert IOR to object` | The probe failed to convert an IOR into an object reference. | Check that the file exists. |
| `Failed to acknowledge alarmIds` | The target system failed to acknowledge the alarms indicated | Check that you specified the correct IDs in the command that you sent to the probe. |
| `Failed to unacknowledge alarmIds` | The target system failed to unacknowledge the alarms indicated | Check that you specified the correct IDs in the command that you sent to the probe. |

## 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 error messages that the probe generates. For information about generic ProbeWatch messages, see the *IBM Tivoli Netcool/OMNIbus Probe and Gateway Guide*.

*Table 15. ProbeWatch messages*

| ProbeWatch message | Description | Triggers/causes |
|---|---|---|
| `ClientSession event loss occurred` | The EMS has failed to push one or more events to the probe. As a result the probe is now out of synchronzation with the EMS. | The EMS indicated that events are being lost and that it is not able to provide the relevant notifications. |
| `ClientSession event loss cleared` | The EMS has restored service and can now provide the relevant events to the probe. On reception of this notification the probe can now resynchronize with the EMS. | The EMS indicated that the event loss period is over and that it is able to provide the relevant notifications once more. |
| `Starting the resynch of the alarm list` | The probe has begun to resynchronize the alarm list. | The occurrence of a resynchronization operation. |
| `Finished resynch of the alarm list` | The probe has completed resynchronizing the alarm list. | The resynchronization of the alarm list completed. |

*Table 15. ProbeWatch messages (continued)*

| **ProbeWatch message** | **Description** | **Triggers/causes** |
|---|---|---|
| `Performing initial resync` | This ProbeWatch message allows you to differentiate between the initial startup resynchronization and the periodic resynchronizations that occur during the normal operation of the probe. | The probe has started up and sent a request for its initial resynchronization of events. |
| `Executing resync command...` | This ProbeWatch message indicates that a `resync` command has been issued. | The probe issued a `resync` command. |
| `Executing resyncFilter command...` | This ProbeWatch message indicates that a `resyncFilter` command has been issued. | The probe issued a `resyncFilter` command. |
| `Performing resync...` | This ProbeWatch message indicates the periodic resynchronization of events that occurs during the normal operation of the probe. | The probe sent a request for a periodic resynchronization. |

# 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.

SC27-6541-04