IBM® Tivoli® Netcool/OMNIbus Generic Probe for TMF814 (V2.1, V3.0 and V3.5) (CORBA) 4.0

Reference Guide
July 28, 2016

SC27-5611-03

Notice

Before using this information and the product it supports, read the information in Appendix A, “Notices and Trademarks,” on page 47.

Edition notice

This edition (SC27-5611-03) applies to version 4.0 of IBM Tivoli Netcool/OMNIbus Generic Probe for TMF814 (V2.1, V3.0 and V3.5) (CORBA) and to all subsequent releases and notifications until otherwise indicated in new editions.

This edition replaces SC27-5611-02.

© Copyright International Business Machines Corporation 2013, 2016.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

About this guide

The following sections contain important information about using this guide.

Document control page

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

The IBM Tivoli Netcool/OMNIbus Generic Probe for TMF814 (V2.1, V3.0 and V3.5) (CORBA) documentation is provided in softcopy format only. To obtain the most recent version, visit the IBM® Tivoli® Information Center:

http://www-01.ibm.com/support/knowledgecenter/?lang=en#!/SSSHTQ/omnibus/probes/common/Probes.html

Table 1. Document modification history

Document version	Publication date	Comments
SC27-5611-00	July 5, 2013	First IBM publication.
SC27-5611-01	November 1, 2013	Updated "Summary" on page 1 to include V2.1 of TMF814 standard, and to show current version number of the probe. Updated "Configuring the probe" on page 2: Updated information on the CORBA connection method and the resynchronization policy. Added entries for Heartbeat policy, Support for Unicode and non-Unicode characters, Command Line Interface, and HTTP/HTTPS command interface. Added "Connecting to the CORBA interface" on page 7, "Configuring the probe" on page 7, and "Connecting to the CORBA interface" on page 7. Added Resynchronization filters to "Alarm retrieval and synchronization" on page 12. Added "Heartbeat" on page 13"Support for Unicode and non-Unicode characters" on page 14. Added "Running the probe" on page 6. Updated "Properties and command line options" on page 19: added **CommandPort**, **CommandPortLimit**, **EncodingStandard**, **HeartbeatInterval**, **NamingContextPath**, **NamingServiceHost**, **NamingServiceIORFile**, **NamingServicePort**, **ResyncProbableCause**, and **ResyncSeverityFilter**; updated **IORFile**,**ORBCharEncoding**, **ORBWCharDefault**, and **ReleaseTMF814**. Updated "Error messages" on page 35 and "Known issues" on page 38. Added Chapter 2, "Migrating from existing probes," on page 39.

© Copyright IBM Corp. 2013, 2016

Table 1. Document modification history (continued)		
Document version	**Publication date**	**Comments**
SC27-5611-02	March 10, 2016	Updated "Summary" on page 1 package version number to 3.0. Added "SSL-based connectivity" on page 5. Added "Properties and command line options provided by the Java Probe Integration Library (probe-sdk-java) version 9.0" on page 26. Updated "Properties and command line options" on page 19. Added descriptions for the following properties: **NotificationClientType**, **ORBDebug**, **ORBDebugFile**, **EnableSSL**, **KeyStore**, **KeyStorePassword**, and **SecurityProtocol**. Updated "Known issues" on page 38 with new section on 32-bit nonnative binary support. Version 3 of the probe addresses the following enhancement requests: • **RFE 53160**: Support for processing a single `StructuredEvent` added.
SC27-5611-03	July 28, 2016	Updated "Summary" on page 1 package version number to 4.0. Added description for the **ResyncAlarmObject** property to "Properties and command line options" on page 19. Descriptions for `$route_cc`, `$SNCState`, `$transmissionParameters`, and `$tpsToModify` added to "Elements" on page 28. Version 4 of the probe addresses the following enhancement requests: • **RFE 82307**: Allow the user to configure the function calls for retrieving active alarms from EMS/NMS. • **APAR IT14347:** Provide support for the following object types: Layered ParameterList, SNC State, Route, and Termination Point Data List in incoming 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`.

Examples of log messages

This guide includes examples of messages that can appear in the probe's log files. These examples omit the time and date stamp for clarity.

Chapter 1. Generic Probe for TMF814

The IBM Tivoli Netcool/OMNIbus Generic Probe for TMF814 (V2.1, V3.0 and V3.5) (CORBA) acquires data from network management systems (NMS) and element management systems (EMS) using a TMF814 (V2.1, V3.0 or V3.5) Common Object Request Broker Architecture (CORBA) interface.

This guide contains the following sections:

Summary

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

The following table summarizes the probe.

Table 3. Summary

Probe target	Telecommunications network devices that comply with TMF814 V2.1, V3.0 or V3.5 standards.
Probe executable name	`nco_p_generic_tmf814`
Package version	4.0
Probe supported on	For details of supported operating systems, see the following Release Notice on the IBM Software Support website: http://www-01.ibm.com/support/docview.wss?uid=swg21653083
Properties file	`$OMNIHOME/probes/`*`arch`*`/generic_tmf814.props`
Rules file	`$OMNIHOME/probes/`*`arch`*`/generic_tmf814.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.

© Copyright IBM Corp. 2013, 2016

Table 3. Summary (continued)	
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*.

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 4. Configuring the probe		
Feature	**Properties**	**See**
Mandatory features:		
CORBA connection method The method that the probe obtains the reference to the object needed to connect to the CORBA interface.	**IORFile** **NamingServiceHost** **NamingServicePort** **NamingServiceIORFile** **NamingContextPath**	“Connecting to the CORBA interface” on page 7
Authentication Credentials for authenticating with the TMF endpoint.	**Password** **Username**	“Authentication” on page 11
TMF release version in use	**ReleaseTMF814**	“Properties and command line options” on page 19
Optional features:		
Resynchronization policy Specifies whether the probe resynchronizes with the TMF endpoint.	**InitialResync** **ResyncInterval** **ResyncBatchSize** **ResyncProbableCauseFilter** **ResyncSeverityFilter**	“Alarm retrieval and synchronization” on page 12
Reconnection policy Specifies whether the probe attempts to reconnect to the TMF endpoint following a communications failure.	**RetryCount** **RetryInterval**	“Reconnection and probe backoff strategy” on page 12
Inactivity policy Specifies whether the probe disconnects from the TMF endpoint following a period of inactivity.	**Inactivity**	“Inactivity” on page 13
Heartbeat policy Specifies whether the probe periodically checks that the connection to the TMF endpoint is still operational.	**HeartbeatInterval**	“Heartbeat” on page 13
Support for Unicode and non-Unicode characters Enables the probe to process alarms that contain characters encoded in UTF-8, such as Asian languages.	**EncodingStandard** **ORBCharEncoding** **ORBWCharDefault**	“Support for Unicode and non-Unicode characters” on page 14

Table 4. Configuring the probe (continued)		
Feature	**Properties**	**See**
Peer-to-peer failover pair 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** **Mode** **PeerHost** **PeerPort** **PidFile** **PropsFile** **RulesFile**	"Peer-to-peer failover functionality" on page 14
HTTP/HTTPS command interface Enables the HTTP/HTTPS command interface and defines the port that it uses.	**NHttpd.EnableHTTP** **NHttpd.ListeningPort** **NHttpd.ExpireTimeout**	"Configuring the command interface" on page 15
Running multiple instances of the probe Allows you to run two or more instances of the probe on a single host machine.	**ORBLocalPort** **Name** **MessageLog** **PidFile** **PropsFile** **RulesFile**	"Running the probe" on page 6

Firewall considerations

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

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

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

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

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

CORBA probes that use EventManager or NotificationManager objects may use different hosts and ports from those that use NamingService and EntryPoint objects. If the probe is configured to get object references from a NamingService or EntryPoint object, you must obtain the host and port information

from the system administrator of the target system. When you have this information, you can add the appropriate firewall rules.

SSL-based connectivity

IBM Tivoli Netcool/OMNIbus Generic Probe for TMF814 (V2.1, V3.0 and V3.5) (CORBA) supports Secure Sockets Layer (SSL) connections. SSL connections provide additional security when the probe retrieves alarms from the target systems.

To enable SSL connections, obtain the required SSL certificates and the Trusted Authority certificate from the IBM Tivoli Netcool/OMNIbus Generic Probe for TMF814 (V2.1, V3.0 and V3.5) (CORBA) server administrator. Add the certificates to a local Java™ keystore so that they can be referenced by the **KeyStore** property.

Prerequisites

The following tools are available to create the keystore:

- 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.
- The Keytool toolkit.

  This is available in the JRE package.

  The keytool can be found at example location:

  `/opt/IBM/tivoli/netcool/platform/linux2x86/jre64_1.7.0/jre/bin/keytool`

Converting the key and certificate into PKCS12 format

If you have a key and a certificate from the server in separate files, you must combine them into a single PKCS12 format file to load into a new keystore. To convert the server certificate into PKCS12 format, use the following OpenSSL toolkit command:

`openssl pkcs12 -export -inkey` *`key_file`* `-in` *`cert_file`* `-out` *`cert_pkcs12`*

Where

`key_file` is the key file retrieved from the server.

`cert_file` is the certificate retrieved from the server.

`cert_pkcs12` is the combined file in PKCS12 format for loading into the keystore.

Creating the SSL keystore

To create a Java keystore, use the following 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 Element Management System (EMS) server, make the following changes to the `generic_tmf814.props` file:

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

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 Generic Probe for TMF814 acquires data from network management systems (NMS) and element management systems (EMS) that implement TMF814 and use a Common Object Request Broker Architecture (CORBA) interface. The probe connects to the NMS or EMS through the CORBA interface and receives events from the device. The probe then processes each event to create an alarm that it forwards on to the ObjectServer.

Data acquisition is described in the following topics:

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
E746578744578743A312E3000000000000010000000000000078000102000000000A3132372E302E31
2E310026250000001F5374616E646172644E532F4E616D655365727665722D504F412F5F726F6F740
000000002000000000000000080000000004A414300000000010000002400000000050100010000000 2
000100010001000F000101090000000205010001000100100
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
000000000078000102000000000A3132372E302E312E310084B80000001F37383136373533383838
2F04221F100E034A10161006304638141414B484C1B0000000002000000000000000800000000 4A41
430000000001000000240000000005010001000000020001000100010001000F000101090000000020501
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:
```

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.

During resynchronization operations, 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`.

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.

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 12 for more information.

To check the connection to the target system, the probe sends a ping command (using the TMF 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 NMS. 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 TMF814 device 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.

You can also use the CLI to manage the probe over an HTTP/HTTPS connection.

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/generic_tmf814 -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:

  ackAlarm
  help
  resync
  resyncFilter
  stop
  unackAlarm
- *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.

ackAlarm

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

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

```
-data '{"command":"ackAlarm", "params":[{"alarmId":"alarmId", "emsId":"emsId",
"managedElementId":"managedElementId", "username":"username"}]}'
```

Where:

- *alarmId* is the identifier stored in the alarm's emsAlarmId field.
- *emsId* is the identifier stored in the alarm's EMS field.
- *managedElementId* is the identifier stored in the alarm's ManagedElement field.
- *username* is the user name of the user acknowledging the alarm (the default is root).

The following example acknowledges the alarm with the following characteristics:

Alarm Identifier: alarm1
EMS Identifier: EMS1
Managed Element Identifier: ME1
Username: root

```
$OMNIHOME/bin/nco_http -uri http://test1.example.com:6789/probes/generic_tmf814
-datatype application/JSON -method POST -data '{"command":"ackAlarm", "params":
[{"alarmId":"alarm1", "emsId":"EMS1", "managedElementId":"ME1",
"username":"root"}]}'
```

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/generic_tmf814
-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/generic_tmf814
-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/generic_tmf814
-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/generic_tmf814
-datatype application/JSON -method POST -data '{"command":"stop", "params":[]}'
```

unackAlarm

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

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

```
-data '{"command":"unackAlarm", "params":[{"alarmId":"alarmId",
"emsId":"emsId", "managedElementId":"managedElementId",
"username":"username"}]}'
```

The parameters have the same meanings as they do for the “ackAlarm” on page 16 command.

The following example "unackanowledges" (clears) the alarm with the same characteristics as the example of the “ackAlarm” on page 16 command:

```
$OMNIHOME/bin/nco_http -uri http://test1.example.com:6789/probes/generic_tmf814
-datatype application/JSON -method POST -data '{"command":"unackAlarm",
"params":[{"alarmId":"alarm1", "emsId":"EMS1", "managedElementId":"ME1",
"username":"root"}]}'
```

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 5. Properties and command line options

Property name	Command line option	Description
EnableSSL *string*	`-noenablessl` (This is equivalent to **EnableSSL** with a value of `false`.) `-enablessl` (This is equivalent to **EnableSSL** with a value of `true`.)	Use this property to specify whether SSL connectivity between the probe and the EMS server is enabled or disabled. This property takes the following values: `true`: SSL connectivity between the probe and the EMS server is enabled. `false`: SSL connectivity between the probe and the EMS server is disabled. The default is `false`.

Table 5. Properties and command line options (continued)		
Property name	**Command line option**	**Description**
EncodingStandard *string*	`-encodingstandard` *string*	Use this property to specify the character encoding standard that the probe uses. Possible values for this property are: `IS0-8859-1`: This sets the encoding standard to Latin Alphabet 1. `UTF-8`: This sets the encoding standard to UTF-8. The default is: `IS0-8859-1`
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. The default is `""`.
KeyStore *string*	`-keystore` *string*	Use this property to specify the location of the keystore file that contains the client certificate for SSL and the trusted authority certificate. The default is `""`.
KeyStorePassword *string*	`-keystorepassword`*string*	Use this property to specify the password required to access the certificate defined in the **Keystore** property. The default is `""`.
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. The default is `""`.

Table 5. Properties and command line options (continued)		
Property name	**Command line option**	**Description**
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. 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. The default is: "".
NamingServicePort *integer*	-nsport *integer*	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. The default is: 0.
NotificationClientType *string*	-notificationclienttype *string*	Use this property to define the probe subscription for notification service. SEQUENCE_EVENT: The probe will receive a sequence of StructuredEvent. STRUCTURED_EVENT: The probe will receive a single StructuredEvent during every notification cycle. The default is: SEQUENCE_EVENT.

Table 5. Properties and command line options (continued)

Property name	Command line option	Description

ORBCharEncoding *string*	`-orncharencoding` *string*	Use this property to specify the native character encoding set that the Oject Request Broker (ORB) uses for character data. Possible values for this property are:
		`IS08859_1`
		`UTF8`
		The default is: `IS08859_1`.

ORBDebug *string*	`-orbdebug` *string*	Use this property to enable ORB debugging. `true`: ORB debugging is enabled. `false`: ORB debugging is disabled. The default is `false`.
ORBDebugFile *string*	`-orbdebugfile` *string*	Use this property to specify the ORB debugging log file. 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. 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. The default is: `0` (the ORB selects a port at random).

ORBWCharDefault *string*	`-orbwchardefault` *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:
		`UCS2`
		`UTF16`
		The default is: `UTF16`.

Table 5. Properties and command line options (continued)		
Property name	**Command line option**	**Description**
Password *string*	`-password` *string*	Use this property to specify the password 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. The default is: `""`.

ReleaseTMF814 *string*	`-releasetmf814` *string*	Use this property to specify the version of TMF that the probe should implement. Possible values for this property are:
		`V2.1`
		`V3.0`
		`V3.5`
		The default is: `V3.0`.

Table 5. Properties and command line options (continued)		
Property name	**Command line option**	**Description**
ResyncAlarmObject *string*	`-resyncalarmobject` *string*	Use this property to specify the objects against which the alarms retrieved during resynchronization are raised. Possible values for this property are: `AllEMSAndME` : All active alarms for EMS itself and all network elements under the control of this EMS are retrieved. This makes the probe request all of the active alarms and threshold crossing alerts (TCAs) that are under the control of the EMS (both those raised by the network elements and those raised by the EMS itself). `AllEMSSystem`: All active alarms for EMS itself only are retrieved, excluding alarms for all network elements managed by this EMS. This makes the probe request all of the active alarms and TCAs for the EMS itself. `AllEMSAndME,AllEMSSystem`: All active alarms for EMS itself and all network elements under the control of this EMS are retrieved, but in a non-TMF-standard way. In contrast to the `AllEMSAndME` property value which is to retrieve the same set of alarms, this property makes probe override the default standard-compliant way. **Note :** This property value caters for whichever EMS implementing getAllEMSAndMEActiveAlarms() operation in a non-TMF-standard way which returns no alarms for EMS itself, but only alarms for network elements being managed. The default is: `AllEMSAndME`.
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`. The default is: `100`.

Table 5. Properties and command line options (continued)		
Property name	**Command line option**	**Description**
ResyncProbableCause Filter *string*	`-resyncpbcausefilter` *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: `pbCause1;pbCause2;pbCause3` 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:
		`PS_INDETERMINATE`
		`PS_CRITICAL`
		`PS_MAJOR`
		`PS_MINOR`
		`PS_WARNING`
		`PS_CLEARED`
		The default is: `""`.
		Note : If the **`AlarmResyncObject`** property is set to `AllEMSSystem`, the **`ResyncSeverityFilter`** property is not used by the probe.

SecurityProtocol *string*	`-securityprotocol` *string*	Use this property to specify the security protocol used. The default is `TLSv1.2`.
StreamCapture *integer*	`-streamcapture` *integer*	Use this property to specify whether the stream capture feature is enabled. The values this property can have are: `1`: The probe uses the stream capture feature. `0`: The probe does not use the stream capture feature. The default is: `0`. **Note :** If you set the value of this property to 1, define a value for the **`StreamCaptureFilePath`** property as well.

Table 5. 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. The default is: "". See "Data stream capture" on page 13 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. 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 6. Properties and command line options		
Property name	**Command line option**	**Description**
CommandPort *integer*	`-commandport` *integer*	Use this property to specify the port to which users can Telnet to communicate with the probe using the Command Line Interface (CLI) supplied. The default is 6970.
CommandPortLimit *integer*	`-commandportlimit` *integer*	Use this property to specify the maximum number of Telnet connections that can be made to the probe. The default is 10.
DataBackupFile *string*	`-databackupfile` *string*	Use this property to specify the path to the file that stores data between probe sessions. The default is " ". **Note :** Specify the path relative to `$OMNIHOME/var`.

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

Table 6. Properties and command line options (continued)		
Property name	**Command line option**	**Description**
MaxEventQueueSize *integer*	`-maxeventqueue size`*integer*	Use this property to specify the maximum number of events that can be queued between the non native process and the ObjectServer. The default is `0`. **Note :** You can increase this number to increase the event throughput when a large number of events is generated.
ResyncInterval *integer*	`-resyncinterval` *integer*	Use this property to specify the interval (in seconds) at which the probe makes successive resynchronization requests. For most probes, the default value for this property is `0` (which instructs the probe to not make successive resynchronization requests). If you are running the JDBC Probe, the default value for the **ResyncInterval** property is `60`. This is because the JDBC Probe only acquires data using the resynchronization process.
RetryCount *integer*	`-retrycount` *integer*	Use this property to specify how many times the probe attempts to retry a connection before shutting down. The default is `0` (which instructs the probe to not retry the connection).
RetryInterval *integer*	`-retryinterval` *integer*	Use this property to specify the length of time (in seconds) that the probe waits between successive connection attempts to the target system. The default is `0` (which instructs the probe to use an exponentially increasing period between successive connection attempts, for example, the probe will wait for 1 second, then 2 seconds, then 4 seconds, and so forth).

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 Generic Probe for TMF814 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 7. Elements specific to events	
Element name	**Element description**
$acknowledgeIndicaton	This element indicates the acknowledgement status of the alarm. The possible values are: • AI_EVENT_ACKNOWLEDGED • 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: 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: • true • 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: true 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. **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 7. 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_AID	This element contains the identifier of the alert associated with this object.
$objectName_EMS	This element contains the name of the EMS that the alarm relates to.
$objectName_Equipment	This element Contains the identification of the equipment that the EMS is part of.
$objectName_EquipmentHolder	This element contains the identification of the equipment holder for the EMS.
$objectName_ManagedElement	This element contains the identifier of the managed element that the alert relates to.
$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.
$route_cc	This element holds the content of all the route_cc elements.
$route_cc_count	This element contains the number of cross-connect entries in the specification of this route.
$route_cc*entry number*_active	This element indicates whether the route is active. This element can take one of the following values: true false .
$route_cc*entry number*_direction	This element indicates the direction of data flow. This element can take one of the following values: UNIdirectional: From the source termination point to the sink termination point. BIdirectional : A two-way communication from and to both termination points.

Table 7. Elements specific to events (continued)

Element name	Element description
$route_cc*entry number*_ccType	This element indicates the type of connection. This element can take the following values: ST_SIMPLE ST_ADD_DROP_A ST_ADD_DROP_Z ST_INTERCONNECT ST_DOUBLE_INTERCONNECT ST_DOUBLE_ADD_DROP ST_OPEN_ADD_DROP ST_EXPLICIT
$route_cc*entry number*_additionalInfo	This element contains a comma-separated list of the route parameters.
$SNCState	This element contains the SNC state of the event. This element can take the following values: SNCS_PENDING: The SNC has been created by an NMS and has not been activated by any NMS; or the SNC has been successfully deactivated by an NMS. SNCS_ACTIVE: The SNC is not in pending state, a route has been assigned to the SNC and all cross-connects for the SNC are active in the network. SNCS_PARTIAL: The SNC is not in pending state, and either a route has not been assigned to the SNC, or not all of the cross-connects of the SNC are active in the network. SNCS_NONEXISTENT: This is used in to report SNCs that have been deleted.
$tpsToModify	This element holds the content of all the tpsToModify elements.
$tpsToModify_tpd_count	This element contains the total number of termination point data items in this termination point data list.
$tpsToModify_tpd*index*_tpName	This element contains the name of the termination point in the form of comma separated of name-value pairs.
$tpsToModify_tpd*index*_tpMappingMode	This element contains mapping mode for the termination point. one of below values: TM_NA TM_NEITHER_TERMINATED_ NOR_AVAILABLE_FOR_MAPPING TM_TERMINATED_AND_AVAILABLE_FOR_MAPPI NG

Table 7. Elements specific to events (continued)	
Element name	**Element description**
$tpsToModify_tpd*index* _transmissionParamsLayersCount	This element contains total number of layers with different rates for the current termination point data.
$tpsToModify_tpd*index* _transmissionParamsLayersRate*index rate number*_paramsCount	This element contains number of transmission parameters for the current termination point data and this named layer.
$tpsToModify_tpd*index* _transmissionParamsLayersRate *number_name*	This element contains the value of this parameter.
$tpsToModify_tpd*index* _ingressTrafficDescriptorName	This element contains an ingress (incoming) traffic descriptor or transmission descriptor in the form of comma separated name-value pairs.
$tpsToModify_tpd*index* _egressTrafficDescriptorName	This element contains an egress (outgoing) traffic descriptor or transmission descriptor in the form of comma separated name-value pairs.
$transmissionParameters	This element holds the content of all the transmissionParameters elements.
$transmissionParameters_layersCount	This element contains the total number of layers with different rates.
$transmissionParameters_layerRate*rate number*_paramsCount	This element contains number of transmission parameters for this named layer.
$transmissionParameters_layerRate*rate number_param name*	This element contains the value of the transmission parameter for the named layer and the named parameter.

Table 8. 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.

Table 8. Elements specific to protection switch events (continued)	
Element name	**Element description**
$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 9. 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.
$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: True (The alarm is a root alarm) 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: • $X733::AdditonalInfo_AlarmNO • $X733::AdditionalInfo_DEVICE_LABEL • $X733::AdditionalInfo_DeviceIP • $X733::AdditionalInfo_Label • $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.

Table 9. Elements specific to alarms (continued)

Element name	Element description
$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: BACKED_UP NOT_BACKED_UP When the attribute has the value BACKED_UP, the value of the $X.733::BackUpObject identifies the back-up object.
$X.733::CorrelatedNotifications	This element contains a list of root alarms that cause correlative alarms. This alarm has the following possible values: • soSource (This field is always empty) • 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: • communicationsAlarm • qualityofServiceAlarm • equipmentAlarm • processingErrorAlarm • securityAlarm • 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 10. 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
`$pmParameterName`	This element contains the `pmParameter` that has crossed the threshold.
`$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 11. Error messages		
Error	**Description**	**Action**
`Alarm alarmID Error in acknowledge.`	The probe could not acknowledge the alarm specified in the **`ackAlarm`** command to 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 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 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.
`Failed to convert IOR to object`	The probe failed to convert an IOR into an object reference.	Check that the value of the **`IORFile`** property is correct and up to date. Also, make sure that the target system is running correctly.

Table 11. Error messages (continued)

Error	Description	Action
`Failed to get active alarms`	The probe received an exception while trying to retrieve active alarms from the EMS manager.	Check the properties related to resynchronization, in particular the filter properties. Check that the target system is running correctly and that there is not an IDL type mismatch.
`Failed to get EMS Manager reference`	The probe is unable to retrieve an EMS Manager reference from the target system.	Check that the target system is running and that there is not an IDL file mismatch.
`Failed to get next iterated alarm batch`	The probe is unable to retrieve the next batch of active alarms.	Check there is not an IDL file mismatch or contact IBM Support.
`Failed to narrow manager reference`	The probe is unable to narrow the EMS Manager reference from the target system.	Check that the target system is running correctly and that there is not an IDL file mismatch.
`Failed to retrieve TMF814 Release`	The probe was unable to retrieve the TMF814 release.	The probe could not retrieve the value of the TMF814 release. Check the value of the **ReleaseTMF814** property and adjust as necessary. In addition, make sure that the properties file has not become corrupted.
`Failed to start`	The probe is unable to start due to an environment problem or an unsupported TMF814 release.	Check that all properties have the correct, and valid, values as defined in "Properties and command line options" on page 19.
`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.
`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.
`Loading TMF814 jar release: UNKNOWN from path` *`file-path`*	The ReleaseTMF814 property has an incorrect, unrecognized value. This message may occur with `Failed to start`.	Ensure the property has one of the values defined in "Properties and command line options" on page 19.

Table 11. Error messages (continued)		
Error	**Description**	**Action**
`Stream capture fails`	The probe was not able to capture raw event data.	Check that the value of the **`StreamCaptureFilePath`** property is a valid file path and tat the path exists.
`Unsupported TMF814 release`	The ReleaseTMF814 property has an incorrect, unrecognized value.	Ensure the property has one of the values defined in "Properties and command line options" on page 19.
`Unable to get EMS session`	The probe is unable to connect to the target EMS session.	Check that the **`Username`** and **`Password`** properties contain the correct values for accessing the target system. In addition, check that the target system is running correctly.
`Unable to ping the EMS session`	The probe is unable to poll the target system during heartbeat checking.	Check that the target system is running correctly.

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 12. 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.	Completion of the resynchronization of the alarm list.

Table 12. ProbeWatch messages (continued)		
ProbeWatch message	**Description**	**Triggers/causes**
`SHUTDOWN. No events received for n seconds`	The probe has shut down due to inactivity.	The probe has not received any events for *n* seconds and so is shutting down. The value of the **`Inactivity`** property determines the value of *n*.

Known issues

Limitation in the HTTP/HTTPS command interface for TMF814 V2.1 devices

The **`ackAlarm`** and **`unackAlarm`** commands of the HTTP/HTTPS command interface are not available for devices that implement TMF814 V2.1. This is due to a limitation in the IDL for that version of TMF814.

Error messages from HTTP/HTTPS command interface

When you use nco_http to send commands, it returns a number of messages. These messages include the following:

```
Warning: W-UNK-103-001: HTTP Server returned an error: 0 Internal Server Error
```

You can ignore this message because the command completes successfully as subsequent messages show.

Misleading response to incorrect HTTP/HTTPS commands

When there are errors in a command sent over the HTTP/HTTPS command interface, a misleading error message is returned. If the command contains the wrong number of parameters, the following message appears in the log file:

```
Information: I-UNK-104-002: {"response":"Parameter set [alarmId] may not be
used with 'unackAlarm", "status":"400"}
```

If a command is missing a single quotation mark (') or a double quotation mark (''), the following messages appear in the log file:

```
Warning: W-ETC-004-060: Undefined argument: params[]
Fatal: F-UNK-102-002: Failed to process arguments (-52: Undefined argument)
```

Null properties

The probe's properties file allows you to set values for the following properties. However, those properties have no effect as the features they correspond to are not implemented in this release of the probe.

`DataBackupFile`
`RotateEndpoint`

Chapter 2. Migrating from existing probes

There are a number of IBM Tivoli Netcool/OMNIbus probes that monitor TMF-814 compliant systems:

- Probe for Alcatel-Lucent NaviScore 9.1
- Probe for Huawei T2000 (CORBA)
- Probe for Nokia-Siemens TNMS (CORBA)
- Probe for ECI Lightsoft (CORBA)

The Generic Probe for TMF814 can also monitor the same systems. This chapter contains guidance on how to migrate from one of the existing probes to the generic probe. The migration procedure has the following stages:

1. Ensure the generic probe contains all the features of the existing probe that you need.
2. Determine any additional features of the generic probe that you want to use.
3. Install the generic probe.
4. Migrate the properties file.
5. Customize the rules file.
6. Run and test the generic probe.
7. Optimize property values and the rules file.

Note : Where possible, carry out the migration in a test environment or a simulation of the production environment so that the work does not interfere with the production environment. Change over to using the Generic Probe for TMF814 in production once you are sure that it behaves in the same way as the probe it is replacing.

Comparison of probe features

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

Common features

The following features are common to all the TMF814 probes:

Table 13. Features common to all TMF814 probes

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

© Copyright IBM Corp. 2013, 2016

Table 13. Features common to all TMF814 probes (continued)	
Functional category	**Features**
Data acquisition	Authentication with the NMS or EMS using a username and password. Ability to receive alarms and notifications. Heartbeat status check. Inactivity timeout. Reconnection and probe backoff. Support for Unicode and non-Unicode characters.

Features specific to the Generic Probe for TMF814

The Generic Probe for TMF814 has the following additional features that are not present in one or more of the product-specific probes:

- "Resynchronization with a probable cause filter" on page 40
- "Resynchronization at intervals" on page 40
- "Data stream capture" on page 40
- "HTTP/HTTPS command interface" on page 40

Resynchronization with a probable cause filter

All TMF814 probes provide a means of resynchronizing with the NMS or EMS using a severity filter. In addition, the generic probe can use a probable cause filter when retrieving alarms, specified by the **ResyncProbableCauseFilter**. "Alarm retrieval and synchronization" on page 12 has more information on resynchronization and the use of filters.

Resynchronization at intervals

In addition to the initial resynchronization with the NMS or EMS, the generic probe also has the ability to resynchronize at regular intervals, specified by the **ResyncInterval** property. "Alarm retrieval and synchronization" on page 12 has more information on resynchronization at intervals.

Data stream capture

The generic probe can capture the stream of binary data and store it in a file specified by the **StreamCapture** and **StreamCaptureFilePath** properties. The data can then be used for debugging purposes, to develop new features, or to pass on to other management systems that require the same data. "Data stream capture" on page 13 has more information on data stream capture.

HTTP/HTTPS command interface

For sites that use Netcool/OMNIbus 7.4 and later, the generic probe has a HTTP/HTTPS command interface. This enables you to send commands to the probe in JSON over a HTTP or HTTPS connection. Commands are available to acknowledge and clear alarms, and perform a resynchronization. "Managing the probe over an HTTP/HTTPS connection" on page 15 has more information on the interface, how to configure it, and the format of the commands. There are also examples of each command.

Features not available in the Generic Probe for TMF814

The Probe for Huawei T2000 (CORBA) has a number of features that are not currently available in the generic probe:

- Event synchronization with the ObjectServer
- EMS Server failover
- Persistence notification

Before deploying the Generic Probe for TMF814 ensure that you do not require any of these features.

Migration procedure

Use this procedure to replace a system-specific probe with the generic probe.

- "Determining the features to use" on page 41
- "Installing the Generic Probe for TMF814" on page 41
- "Migrating properties" on page 41
- "Customizing the rules file" on page 43
- "Running and testing the probe" on page 46
- "Optimizing property values and the rules file" on page 46

Determining the features to use

Determine which features of the Generic Probe for TMF814 you require to adequately replace the system-specific probe.

Use the information in Chapter 1, "Generic Probe for TMF814," on page 1 and in "Comparison of probe features" on page 39 to determine the features that you want to implement in the generic probe. Start by determining the features that provide the functionality that the system-specific probe gives you. Then, decide which of the new features specific to the generic probe you want to use, if any. Take careful note of the requirements of each feature and the information you need to implement them.

Installing the Generic Probe for TMF814

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

Migrating properties

Determine the values required for the properties file of the generic probe. As shown in "Configuring the probe" on page 2 there are two groups of properties:

- Required settings that all installations of the probe must have.
- Optional settings that related to features that are not mandatory.

Use the properties file for the system-specific probe to set the correct values in the generic probe.

Required settings

The required settings relate to the following:

- CORBA connection method
- Authentication
- TMF release version in use

There are two ways of connecting to the CORBA interface, as shown in "Connecting to the CORBA interface" on page 7. From the implementation in the system specific probe, determine which you want to use on the generic probe and set the properties accordingly. Note that the names of properties for using a Naming Service to establish the connection differ in the generic probe, as the following table shows:

Table 14. CORBA connection properties	
System-specific probe	**Generic probe**
ORBInitialHost	**NamingServiceHost**
ORBInitialPort	**NamingServicePort**
NamingContextIORFile	**NamingServiceIORFile**

Copy the values for the authentication properties from the system-specific properties file to the file for the generic probe.

The **ReleaseTMF814** property is found only in the generic probe. Set this property to the version of the TMF814 standard that the NMS or EMS implements.

Optional settings

Optional settings relate to features of the probe that a site may decide to use or not, as their business requires:

- Resynchronization policy
- Reconnection policy
- Inactivity policy
- Heartbeat policy
- Support for Unicode and non-Unicode characters
- Peer-to-peer failover pair
- HTTP/HTTPS command interface
- Running multiple instances of the probe

The table in "Configuring the probe" on page 2 lists the properties associated with each feature. For features that you use in the system-specific probe, set the values of the generic probe properties to appropriate values. For features that you do not use in the system-specific probe, or that the system-specific probe does not provide, decide on the values to use and set the generic properties file accordingly. Follow the advice in the sections referenced in the configuration table for details of how to set the properties.

Note that the generic probe uses different names for some properties to those used in the system-specific probes, as the following table shows:

Table 15. Properties with different names in the generic probe	
System-specific property	**Generic property**
AgentHeartbeat	**HeatbeatInterval**
ExcludeSeverityCleared **ExcludeSeverityCritical** **ExcludeSeverityIndeterminate** **ExcludeSeverityMajor** **ExcludeSeverityMinor** **ExcludeSeverityWarning**	**ResyncSeverityFilter**
Resynch	**InitialResync** **ResyncInterval**
ResynchBatchSize	**ResyncBatchSize**
Retry	**RetryCount** **RetryInterval**

Table 15. Properties with different names in the generic probe (continued)	
System-specific property	**Generic property**
Timeout	**Inactivity**

In particular notice that there is no direct correspondence between the properties that filter alarms by severity. From the system-specific probe's properties file, determine the severities that you want to filter and then construct the appropriate value for the **ResyncSeverityFilter** property of the generic probe. “Properties and command line options” on page 19 defines the format for the value of the **ResyncSeverityFilter** property.

Customizing the rules file

Edit the rules file for the generic probe to:

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

Note : The generic probe may not be able to parse certain attributes if the vendor does not follow the TMF814 standard or has implemented their own types that are not TMF814 compliant.

Attributes

There are some differences in the names or values of attributes between the system-specific probes and the generic probe. The following table indicates where there are differences, and shows the element that the TMF814 standard defines. Be sure to make the necessary changes if you copy over rules from the system-specific rules file.

Table 16. Differences in rules file attributes		
TMF814 element name	**System-specific probes**	**Generic probe**
`event_name`	`$event_name`	`$EventName`
`event_type.domain_name`	`$domain_name`	`$DomainName`
`objectName`	`$`*name* For example: `$AID`, `$EMS`, `$ManagedElement`.	`$objectName_`*name* For example: `$objectName_AID` `$objectName_EMS` `$objectname_ManagedElement`
`groupName`	`$`*name*	`$groupName_`*name*
`acknowledgeIndication`	Not available in the system-specific probes.	`$acknowledgeIndication`
`attributeList`	`$attributeList= "`*attributes*`;"` Here, *attributes* is a list of attribute/value pairs with a comma separating each pair. For example: `$attributeList="a=1, b=2, c=3;"`	`$attributeList_`*name*`= "`*value*`"` For example: `attributeList_a="1"` `attributeList_b="2"` `attributeList_c="3"`

Table 16. Differences in rules file attributes (continued)

TMF814 element name	System-specific probes	Generic probe
protectedTP	$protectedTP= "*terminationpoints*;" Here, *terminationpoints* is a list of termination points and their values separated by commas. For example: $protectedTP="a=1, b=2, c=3;"	$protectedTP_*name*="*value*" For example: $protetectedTP_a="1" $protectedTP_b="2" $protectedTP_c="3"
switchToTP	$switchToTP= "*terminationpoints*;" Here, *terminationpoints* is a list of termination points and their values separated by commas. For example: $switchToTP="a=1, b=2, c=3;"	$switchToTP_*name*="*value*" For example: $switchToTP_a="1" $switchToTP_b="2" $switchToTP_c="3"
switchAwayFromTP	$switchAwayFromTP= "*terminationpoints*;" Here, *terminationpoints* is a list of termination points and their values separated by commas. For example: $switchAwayFromTP="a=1, b=2, c=3;"	$switchAwayFromTP_*name*= "*value*" For example: $switchAwayFromTP_a="1" $switchAwayFromTP_b="2" $switchAwayFromTP_c="3"
switchAwayFromE	$switchAwayFromE= "*equipmentlist*;" Here, *equipmentlist* is a list of equipment attributes and their values separated by commas. For example: $switchAwayfromE="a=1, b=2, c=3;"	$switchAwayFromE_*name*= "*value*" For example: $switchAwayFromE_a="1" $switchAwayFromE_b="2" $switchAwayFromE_c="3"
switchToE	$switchToE= "*equipmentlist*;" Here, *equipmentlist* is a list of equipment attributes and their values separated by commas. For example: $switchToE="a=1, b=2, c=3;"	$switchToE_*name*="*value*" For example: $switchToE_a="1" $switchToE_b="2" $switchToE_c="3"
protectedE	$protectedE= "*equipmentlist*;" Here, *equipmentlist* is a list of equipment attributes and their values separated by commas. For example: $protectedE="a=1, b=2, c=3;"	$protectedE_*name*="*value*" For example: $protectedE_a="1" $protectedE_b="2" $protectedE_c="3"

Table 16. Differences in rules file attributes (continued)

TMF814 element name	System-specific probes	Generic probe
performanceValue	Not available in the system-specific probes.	$performanceValue_pmParameter $performanceValue_pmLocation $performanceValue_intervalStatus $performanceValue_unit $performanceValue_value
thresholdValue	Not available in the system-specific probes.	$thresholdValue_pmParameter $thresholdValue_pmLocation $thresholdValue_intervalStatus $thresholdValue_unit $thresholdValue_value
affectedTPList	$affectedTPList="*terminationpoints*;" Here, *terminationpoints* is a list of termination point names and their values separated by commas. For example: $affectedTPList="a=1, b=2, c=3;"	$affectedTPList_*name*="*value*" For example: $affectedTPList_a="1" $affectedTPList_b="2" $affectedTPList_c="3"
additionalInfo	$*name*	$additionalInfo_*name*
X.733::SpecificProblems	$X.733::SpecificProblems*n* Here, *n* is an integer greater than or equal to zero.	$X.733::SpecificProblems_*name*
X.733::BackUpObject	$*name*	$X.733::BackUpObject_*name*
X.733::Correlated Notifications	$X.733::Correlated Notifications*n*= "Sources: source_name*n*= source_value*n*,...; NotificationIds: notifId*n*,...;" Here, *n* is an integer greater than or equal to zero.	$X.733::Correlated Notifications_*n*= "Sources: source_name*n*= source_value*n*,...; NotificationIds: notifIf*n*,...;"
X.733::MonitoredAttributes	$*name*	$X.733::Monitored Attributes_*name*
X.733::ProposedRepair Actions	$X.733::ProposedRepair Actions*n*	$X.733::ProposedRepair Actions_*n*
X.733::AdditionalInfo	$*name*	$X.733::AdditionalInfo_*name*

Running and testing the probe

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

To run and test the probe:

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

   ```
   $OMNIHOME/probes/nco_p_generic_tmf814 -messagelog stdout -messagelevel debug
   -initialresync true
   ```

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

   ```
   Information: I-JPR-000-000: Probe connected
   ```

   If the probe fails to connect:

   - Check and adjust the properties related to setting up a connection. See "Connecting to the CORBA interface" on page 7 for information on the connection properties and how to set them.
   - Ensure that any firewall between the probe host and the NMS or EMS is configured to allow traffic to pass from one end to the other in both directions.

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

   ```
   Debug: D-JPR-000-000: Send request for active alarms
   Debug: D-JPR-000-000: Successfully retrieved batch of 100 resync alarms.
   Information: I-JPR-000-000: Parsing alarm
   Debug: D-JPR-000-000: Resync alarms will be returned in batches, using iterator
   Debug: D-JPR-000-000: Fetching batch of alarms
   ```

   Troubleshoot any synchronization errors, including the values of the synchronization properties. See "Alarm retrieval and synchronization" on page 12 for information on synchronization.

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

   ```
   Debug: D-UNK-000-000: [Event Processor] nativeProbableCause:       HP_REI
   Debug: D-UNK-000-000: [Event Processor] neTime:    20120626071005.0Z
   Debug: D-UNK-000-000: [Event Processor] EventName:
   Debug: D-UNK-000-000: [Event Processor] notificationId:
   Debug: D-UNK-000-000: [Event Processor] perceivedSeverity: 1
   Debug: D-UNK-000-000: [Event Processor] DomainName:        tmf_mtnm
   Debug: D-UNK-000-000: [Event Processor] probableCauseQualifier:    1-79
   Debug: D-UNK-000-000: [Event Processor] objectTypeQualifier:
   Debug: D-UNK-000-000: [Event Processor] layerRate: 15
   ```

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

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

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

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

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

Optimizing property values and the rules file

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

Appendix A. Notices and Trademarks

This appendix contains the following sections:

- Notices
- Trademarks

Notices

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

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

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

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

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

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

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

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

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

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

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

IBM Corporation
Software Interoperability Coordinator, Department 49XA

© Copyright IBM Corp. 2013, 2016

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.

SC27-5611-03