IBM® Tivoli® Netcool/OMNIbus Probe for Nokia Network Functions Manager for Packet 2.1

Reference Guide
December 14, 2022

SC27-8758-06

Notice

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

Edition notice

This edition (SC27-8758-06) applies to version 2.1 IBM Tivoli Netcool/OMNIbus Probe for Nokia Network Functions Manager for Packet (NFM-P) and to all subsequent releases and modifications until otherwise indicated in new editions.

This edition replaces SC27-8758-05.

Contents

About this guide

The following sections contain important information about using this guide.

Document Control Page

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

The IBM Tivoli Netcool/OMNIbus Probe for Nokia Network Functions Manager for Packet (NFM-P) documentation is provided in softcopy format only. To obtain the most recent version, visit the IBM® Tivoli® Netcool® Knowledge Center:

https://www.ibm.com/support/knowledgecenter/SSSHTQ_int/omnibus/common/kc_welcome-444.html?lang=en

Table 1. Document modification history

Document version	**Publication date**	**Comments**
SC27-8758-00	July 20, 2017	First IBM publication.
SC27-8758-01	August 9, 2018	Updated for version 2.0 of the probe. "Summary" on page 1 updated. The Probe for Nokia Network Functions Manager for Packet has been certified to run against Nokia NSP 18.6.
SC27-8758-02	June 28, 2019	"Summary" on page 1 updated. The Probe for Nokia Network Functions Manager for Packet has been certified to run against Nokia NSP 19.3.
SC27-8758-03	April 3, 2020	"Summary" on page 1 updated. The Probe for Nokia Network Functions Manager for Packet has been certified to run against Nokia NSP 19.11.
SC27-8758-04	July 24, 2020	"Summary" on page 1 updated. The Probe for Nokia Network Functions Manager for Packet has been certified to run against Nokia NSP 20.6.
SC27-8758-05	September 30, 2021	"Summary" on page 1 updated. Support has been extended to version 21.x and the probe has been certified to run against Nokia NSP 21.6. "Configuring Java" on page 5 updated.
SC27-8758-06	December 15, 2022	Updated for version 2.1 of the probe. "Summary" on page 1 updated. The probe has been certified to run against Nokia NSP 22.6. "Known Issues" on page 42 updated.

Conventions used in this guide

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

Operating system-dependent variables and paths

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

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

```
$OMNIHOME/probes
```

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

```
%OMNIHOME%\probes
```

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

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

Operating system-specific directory names

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

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

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

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

Table 2. Directory names for the arch variable

Operating system	Directory name represented by *arch*
AIX® systems	`aix5`
Red Hat Linux® and SUSE systems	`linux2x86`
Linux for System z	`linux2s390`
Solaris systems	`solaris2`
Windows systems	`win32`

OMNIHOME location

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

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

Chapter 1. Nokia Network Functions Manager for Packet

The Nokia Network Functions Manager for Packet (NFM-P) is the Network Services Platform (NSP) module for IP/MPLS management. It enables end-to-end network and service management across all domains of a converged, all-IP network. NFM-P helps service providers maximize operational efficiencies through provisioning, troubleshooting, and proactive assurance.

The Probe for Nokia Network Functions Manager for Packet acquires data from Nokia NFM-P using a Java™ Messaging System (JMS).

Note: Nokia NFM-P was previously known as 5620 Service Aware Manager (SAM).

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 provides a summary of the Probe for Nokia Network Functions Manager for Packet.

Table 3. Summary

Probe target	The probe supports all revisions of Nokia NFM-P Release 17.x, 18.x, 19.x, 20.x, and 21.x (x denotes 3,6,9 and 11). **Note:** The Probe for Nokia Network Functions Manager for Packet has been certified to run against Nokia NSP 17.3, 18.6, 19.3, 19.11, 20.6, 21.6, and 22.6.
Probe executable names	`nco_p_nokia_nfmp`
Probe installation packages	`omnibus-`*`arch`*`-probe-nco-p-nokia-nfmp-`*`version`*
Package version	2.1

Table 3. Summary (continued)

Probe supported on	For details of supported operating systems, see the following Release Notices on the IBM Software Support website: http://www-01.ibm.com/support/docview.wss?uid=swg22005197
Properties files	`$OMNIHOME/probes/`*`arch`*`/nokia_nfmp.props` Details about storing commands in this file are described in "Storing commands in the nco_http properties file" on page 22.
Rules files	`$OMNIHOME/probes/`*`arch`*`/nokia_nfmp.rules`

Alarm pre-classification configuration files	`AdvCorr36.include.compat.rules`
	
	`nokia_nfmp-preclass.include.snmptrap.rules`
	
	`nokia_nfmp-preclass.snmptrap.lookup`
	
	These configuration files are used by the probe for alarm pre-classification. For details see "Configuring pre-classification" on page 6.

SQL files	`advcorr.sql` Details about running this SQL file are described in "Configuring the ObjectServer to support intra-device correlations" on page 7.

Lookup file	`nokia_nfmp.lookup`
	
	This lookup file is described in "Configuring lookup tables" on page 6.
	
	`modify_dedup_trigger.sql`
	
	This sql file is described in "Known Issues" on page 42 under the Modifying the default Netcool/OMNIbus deduplication triggers heading.

Advanced JMS filter XML file	registerNotification.txt The Nokia NFM-P server uses this file to determine which events to send to the probe when the **`JmsTopic`** property is set to `5620-SAM-topic-xml-filtered`. It is specified by the **`AdvancedFilterConfigFile`** property of the `nokia_nfmp.props` file, for details see "Properties and command line options" on page 22.
environment file	`$OMNIHOME/probes/java/nco_p_nokia_nfmp.env` Configuring the environment file is described in "Jar files" on page 5.
Requirements	For details of any additional software that this probe requires, refer to the `description.txt` file that is supplied in its download package.
Connection method	Java Messaging System (JMS) and HTTP(s) Port (optional)

Table 3. Summary (continued)

Remote connectivity	The Probe for Nokia Network Functions Manager for Packet can connect to a remote device. Details of the remote device are specified using the **Host**, **EJBPort**, and **HTTPPort** properties in the properties file; for descriptions of these properties, see "Properties and command line options" on page 22.
Multicultural Support	Not Available
Peer-to-peer failover functionality	Available
IP environment	IPv4 or 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 IBM Documentation:

   https://www.ibm.com/support/knowledgecenter/SSSHTQ_int/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 in IBM Documentation:

   https://www.ibm.com/support/knowledgecenter/SSSHTQ_int/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.

This section contains topics on the Nokia NFM-P configuration requirements:

- "Configuring Java" on page 5

Configuring the cross-launch application

The probe is supplied with a launch-in-context feature that enables you to launch the Nokia NFM-P Web app from the Netcool/OMNIbus Web GUI Event Viewer right-click tool. Users can configure the GUI manually.

Configure the Web GUI manually

To configure the Web GUI manually, complete the following steps.

1. Edit the object server alerts.status table to add ObjectFullName field using sql file addObjectFullName.sql

   For example:

   ```
   $OMNIHOME/bin/nco_sql -server NCOMS -username root -password
    '' </opt/IBM/tivoli/netcool/omnibus/probes/
   linux2x86/NOKIA_NFMP_CrossLaunch/addObjectFullName.sql
   ```

2. Login into the IBM Tivoli Netcool/OMNIbus WebGUI Dashboard Application Service Hub.

   https://<DASH IP or hostname>:16311/ibm/console

   Refer to the *IBM Tivoli Netcool/OMNIbus Web GUI Administration and User's Guide* to create a new tool of script type named `NFMP-CrossLaunch` with the following inputs.

   For example:

   a. Open the Event Management Tool (from the Dashboard's side bars)

   b. Select 'Tool Configuration' and create a new tool named "NFMP-CrossLaunch"

   c. Choose type as script and input script as below:

   ```
   var objName='{@ObjectFullName}'; var samhost = '127.0.0.1';
   var address = 'http://' + samhost + '/FaultManagement?
   view=alarmListImpacts&objectFullName='+ encodeURIComponent(objName);
   window.open (address,"NFM-P Cross Launch");
   ```

   d. * replace the IP <127.0.0.1> with your Nokia NSP server hostname or IP address accordingly.

   Note: When SSL is enabled on NSP, ensure the following settings are correct:

   i) Ensure the protocol is changed from `http` to `https`.

   ii) Ensure the IP address contains a port number configured for the NSP Client; for example, `127.0.0.1:4321`

3. Refer to *IBM Tivoli Netcool/OMNIbus Web GUI Administration and User's Guide* to perform the menu configuration and modify alert menu to include NFMP-CrossLaunch that should be created in step 2 from available items.

   For example:

   a. **Open Event Management Tool** > **Menu Configurations**

   b. Click on "alerts" from the available menu and click on the "Modify" button.

c. Add the **NFMP-CrossLaunch** tool that was previously created in step 2 from the available items to the current items.
d. Click **Save**.
e. Open **Event Viewer**.
f. Click **Refresh**.
g. Right click on one of the alarms received from NFM-P. You should see **NFMP-CrossLaunch** as an option.

Configuring Java

Before running the probe, you should configure your PATH and `JAVA_HOME` environment variable for the probe to use JRE OpenJDK 11.0.12.

To do so on UNIX and Linux operating systems, run the following command:

```
export PATH=/usr/bin/jre8/jre/bin:$PATH
```

```
export JAVA_HOME=/usr/bin/jre8/jre
```

To do so on Windows operating systems, run the following command:

```
set PATH=C:\\Java\\jre8\\jre\\bin;%PATH%
```

```
set JAVA_HOME=C:\\Java\\jre8\\jre
```

Note: The probe has been tested against the following environment:

IBM Semeru Runtime Open Edition: 11.0.12.0 (build 11.0.12+7)

Eclipse OpenJ9 VM 11.0.12.0 (build openj9-0.27.0, JRE 11 Linux amd64-64-Bit Compressed References 20210729_228 (JIT enabled, AOT enabled)

OpenJ9: 1851b0074

OMR: 9db1c870d

JCL: 21849e2ca0 based on jdk-11.0.12+7

Jar files

As a part of the configuration process, you must copy the `samOss.jar` file to the `java` folder in your Netcool/OMNIbus installation.

On UNIX and Linux operating systems, the `java` folder is in the following location:

```
$OMNIHOME/probes/java
```

On Windows operating systems, the `java` folder is in the following location:

```
%OMNIHOME%\probes\win32
```

Configuring the NFMPJARHOME environment variable

If you want to use a path other than the default path mentioned in the previous section for the `samOss.jar` file, you must configure the `NFMPJARHOME` environment variable accordingly.

On UNIX and Linux operating systems, configure the `$NFMPJARHOME` environment variable in the probe environment file (`$OMNIHOME/probes/java/nco_p_nokia_nfmp.env`) to include the path to `samOss.jar`. To do this, add the following line:

```
NFMPJARHOME=directory_path
```

Where *`directory_path`* is the full path to `samOss.jar` file

For example, add the line:

```
NFMPJARHOME=/opt/nfmp/samOss.jar
```

Note: As an alternative to setting the NFMPJARHOME environment variable, copy the samOss.jar file to ${OMNIHOME}/probes/java and the probe will load the Jar file from there.

On Windows operating systems, configure the %NFMPJARHOME% environment variable to include the path to samOss.jar. To do this, run the following command:

```
set NFMPJARHOME=directory_path
```

Where *directory_path* is the full path to samOss.jar file

For example, run the following command:

```
set NFMPJARHOME=C:\nfmp\jar
```

Note: As an alternative to setting the NFMPJARHOME environment variable, copy the samOss.jar file to %OMNIHOME%\probes\win32 and the probe will load the Jar file from there.

Location of the jar files

If you use the default installation, you can find the jar file in the following location of the Nokia NFM-P server:

```
/opt/nsp/nfmp/server/nms/integration/SAM_O/samOss.jar
```

Configuring lookup tables

The probe is supplied with a lookup table that contains details of the various types of alarms that Nokia NFM-P generates. You may need to update the rules file to include the path to the lookup table in your installation.

The following lookup file is delivered with the Probe for Nokia Network Functions Manager for Packet:

```
nokia_nfmp.lookup
```

This file is installed in the following location:

```
$OMNIHOME/probes/includes
```

The path to the lookup file is included in the rules file. If you are not running the probe from $OMNIHOME/probes, you must change this path accordingly.

Note: You cannot use $OMNIHOME in the paths to the lookup files.

Configuring pre-classification

In addition to automated deduplication and generic-clear event correlation, this probe also supports alarm pre-classification. Pre-classification allows you to identify and flag within the probe rules file the causal relevance of each event type. This feature enables you to overcome many of the shortcomings of Root Cause Analysis (RCA) systems by more efficiently and accurately determining the root cause of a failure from the events processed.

Advanced correlation is implemented by building a lookup table referenced by the probe rules file to determine the causal relevance of a received event before it is forwarded to the Netcool/OMNIbus ObjectServer.

Details about the causal relevance of each event type are stored in the nokia_nfmp-preclass.snmptrap.lookup file. You can modify this file to add to or change the causal relevance details.

The entry for each event type uses the following format:

```
{ "Event_Id","Causal_relevance" }
```

For example: `{"Nokia-NSP-NFM-P-LinkDown","0"},`

The causal relevance of each event type is defined using one of the of following predefined integer values:

0 – Unknown

1 – Root Cause

2 – Symptom

3 – Singularity

4 – Information

To determine the causal relevance assigned to an entry, use the following guidelines:

Unknown
If the condition represented by an event cannot be classified as a root cause, symptom, singularity, or informational event, it is Unknown.

Root Cause
A condition that is known not to be caused by any other detectable condition.

Symptom
A condition that is known to always be caused by another detectable condition. Generally these are events that indicate degraded conditions or failures of higher level entities or processes.

Singularity
If an event represents a degraded condition or failure that is known to not be a root cause, and cannot be caused by another condition, it is a singularity event.

Information
If an event provides information about a system that does not represent a degraded condition or failure, or it indicates the clearing, or resolution, of a previously occurring fault related condition, it is an informational event.

To use the alarm pre-classification feature provided with this probe, you must perform the tasks described in the following topics:

- "Configuring the ObjectServer to support intra-device correlations" on page 7
- "Manually adding conversions to the ObjectServer" on page 8

Configuring the ObjectServer to support intra-device correlations

You must perform this task as an IBM Tivoli Netcool/OMNIbus user with ISQLWrite permissions.

From a command prompt, run the extracted `advcorr.sql` script using one of the following platform-dependent, case-sensitive commands.

UNIX and Linux operating systems

On UNIX and Linux operating systems, run the following command:

```
$OMNIHOME/bin/nco_sql -server objectserver_name -user username -password
password < path_to_file/advcorr.sql
```

where:

`$OMNIHOME` is your installation location of Tivoli Netcool/OMNIbus.

`objectserver_name` is the name assigned to your ObjectServer.

`username` and *`password`* are your ObjectServer login details.

`path_to_file` is the path to the `advcorr.sql` file.

Windows operating systems

On Windows operating systems, run the following command:

```
%NCHOME%\bin\redist\isql.exe -S objectserver_name -U username -P password -i
path_to_file\advcorr.sql
```

where:

`%NCHOME%` is your installation location of IBM Tivoli Netcool/OMNIbus.

`objectserver_name` is the name assigned to your ObjectServer.

`username` and *`password`* are your ObjectServer login details.

`path_to_file` is the path to the `advcorr.sql` file.

Messages generated

As part of the configuration, the script attempts to drop intra-device correlation tables (and associated triggers) which might have been created during a previous installation. As this is a first-time installation, no such tables or triggers exist, and an error listing is generated on completion. These messages are harmless and can be ignored. Sample output is shown below:

```
ERROR=Object not found on line 102 of statement
'--#######################################################################
######...', at or near 'AdvCorr_SetCauseType'
ERROR=Object not found on line 1 of statement 'drop trigger
AdvCorr_LPC_RC;...',
at or near 'AdvCorr_LPC_RC'
ERROR=Object not found on line 1 of statement 'drop trigger
AdvCorr_LPC_Sym;...', at or near 'AdvCorr_LPC_Sym'
ERROR=Object not found on line 4 of statement '-- Drop tables in case they
already exist from a previous installation...', at or near
'AdvCorrLpcSymCand'
ERROR=Object not found on line 1 of statement 'drop table
alerts.AdvCorrLpcRcCand;...', at or near 'AdvCorrLpcRcCand'
(0 rows affected)
(0 rows affected)
(0 rows affected)
(0 rows affected)
(0 rows affected)
(0 rows affected)
(0 rows affected)
(0 rows affected)
(0 rows affected)
(0 rows affected)
(0 rows affected)

(0 rows affected)
(0 rows affected)
(0 rows affected)
(0 rows affected)
```

Objects created in the ObjectServer

The `advcorr.sql script` creates the following objects in the ObjectServer to aid in determining the causal relevance of events:

- Intra-device correlation (`AdvCorr`) tables within the alerts database
- Supplementary automations implemented as an `AdvCorr` trigger group and three related triggers
- Additional columns in the `alerts.status` table

Manually adding conversions to the ObjectServer

Conversions are required to support the two columns (`AdvCorrCauseType` and `CauseType`) that were added to the `alerts.status` table as a result of the ObjectServer configuration (see "Configuring the ObjectServer to support intra-device correlations" on page 7). These conversions translate Type integer values (0 - 4) into descriptive causal relevance text for display within the event list.

Note: If you are using Netcool/OMNIbus Knowledge Library version 3.8 or above, these conversions are automatically created so you will not need to add them manually.

To add the required conversions, use the following steps:

1. From the Tivoli Netcool/OMNIbus Administrator window, click the **Visual** dropdown list, and then click the **Conversions** icon.
2. Right click **alert.status** and click **Add Conversion** to access the **Conversion Details** window.
3. Select **AdvCorrCauseType** from the **Column** dropdown list, and then create the following entries within the **Value** and **Conversion** fields, clicking **OK** to save each set of entries:

Table 4. Value and conversion mapping

Value	**Conversion**
0	Unknown
1	Root cause
2	Symptom
3	Singularity
4	Information

4. Repeat steps 2 and 3 to set up the same conversions for the `CauseType` column, substituting `CauseType` as the **Column** field entry in step 3.

   On completion, the **Conversions** window appears.
5. Enable the pre-class rules by uncommenting the following lines found in the `nokia_nfmp.rules` file:

```
include "../includes/nokia_nfmp-preclass.include.snmptrap.rules"
include "../includes/AdvCorr36.include.compat.rules"
```

Note: Without the configuration described in this topic, the probe will only use the default rules file configuration and will ignore the alarm pre-classification processing.

Establishing a secure connection between the probe and Nokia NFM-P

To establish a secure communication between the probe and the Nokia NFM-P interface, you should use an SSL connection with HTTPS. Such a connection significantly reduces the vulnerabilities of hijacked sessions and passwords.

Configuring SSL connections

If the Nokia NFM-P server is using a Secure Socket Layer (SSL) connection to encrypt data exchanged over JMS and HTTP, you will need to configure the truststore for the HTTPS connection on the Netcool/OMNIbus probe server.

To configure the truststore, use the following steps:

1. Obtain the security certificate from the NFM-P server.
2. Import the security certificate from the NFM-P server.
3. Verify that the security certificate has been imported into the keystore.

Obtaining a certificate file into the truststore

There are two possible approaches:

1. Obtaining Nokia NFM-P security certificate from certificate authority (CA)

2. Exporting security certificate file from an existing keystore file from NFM-P server using the command:

   ```
   ./keytool -export -alias alias_name -keystore keystore_file -storepass
   password -file certificate_file
   ```

   Where:

   alias_name is the keystore alias specified during Nokia NFM-P keystore generation, for example: NFMP_ALIAS.

   keystore_file is the path to and name of the Nokia NFM-P keystore file, for example: `/opt/nfmpserver.keystore`.

   password is the Nokia NFM-P keystore password, for example: the password of `nfmpserver.keystore`.

   certificate_file is the path to and name of the certificate file to be created, for example: `/opt/nfmpcert`.

Importing a security certificate into a new or an existing truststore on the Netcool/OMNIbus probe server

To import a certificate file into the truststore, use one of the following steps:

1. For importing the certificate into a new truststore, use the following command:

   ```
   ./keytool -import -trustcacerts -alias new_alias_name -file certificate_file
   -keystore truststore_file -storepass password
   ```

   Note: If the alias does not point to an existing key entry in a truststore file, then keytool assumes you are adding a new trusted certificate entry into truststore file. In this case, the alias should not already exist, otherwise importing fails.
2. For importing the certificate into an existing truststore, use the following command:

   ```
   ./keytool -import -trustcacerts -alias alias_name -file certificate_file
   -keystore truststore_file -storepass password
   ```

   Note: If the alias points to a key entry in a truststore file, then keytool assumes you are importing a certificate reply, replacing old certificate chain with new certificate chain in truststore file.

   Where:

   alias_name is the key entry of the certificate reply. The alias must be the same as that specified during keystore file generation in Nokia NFM-P server, for example: NFMP_ALIAS.

   new_alias_name is the keystore alias of a new keystore, for example: NFMP_ALIAS_NEW.

   certificate_file is the path to and name of the certificate file created earlier, for example: `/opt/nfmpcert`.

   truststore_file is the path to and name of the truststore file that will contain the imported certificate, for example: `/opt/nfmpserver.truststore`.

   password is the Nokia NFM-P keystore password, for example: the password of `nfmpserver.truststore`.

Verifying that the security certificate has been imported into the keystore

To verify that the certificate has been imported into the keystore, use the following command:

```
./keytool -list -v -keystore truststore_file
```

Where:

`truststore_file` is the path to and name of the truststore file generated, for example: `/opt/nfmpserver.trustStore`.

Note: For more details about configuring SSL security for the Nokia NFM-P server (including instructions about obtaining certificate files) refer to the NFM-P Installation and Upgrade Guide.

Configuring the probe

To configure the probe to connect to the Nokia NFM-P server using an SSL connection, use the following steps:

1. Set the probe's **`UseSSL`** property to `true` and configure the probe's **`HTTPPort`** property to use the default Nokia NFM-P HTTPS port, 8443.
2. Specify values for the following probe properties:
   - **`TrustStore`**: Specify the path of the probe's Java keystore that you created in the steps for importing the certificate into the truststore, for example: `/opt/nfmpserver.trustStore`.
   - **`TrustStorePassword`**: Specify the password that you set for the Java keystore.
   - **`CertificateStore`**: Specify the path of the certificate keystore. This will be the same value as that set for the **`TrustStore`** property unless you manage them in different keystore files.
   - **`CertificateStorePassword`**: Specify the password set for the certificate keystore.

You can set both the **`TrustStore`** property and the **`CertificateStore`** property to the same keystore file where the license file is imported, or you can specify different keystore files.

Example SSL configuration property settings

The following example shows SSL configuration settings from the properties file of an example Probe for Nokia Network Functions Manager for Packet:

```
Host                       : "198.162.20.21"
HTTPPort                  : 8443
UseSSL                    : "true"
TrustStore                  : "/opt/nfmpserver.trustStore"
TrustStorePassword          : "newpassword"
CertificateStore          : "/opt/nfmpserver.trustStore"
CertificateStorePassword    : "newpassword"
```

Displaying unicode and non-unicode characters

The probe can support multibyte characters and so can display both unicode and non-unicode characters.

Before using the probe to process data that contains multibyte characters, perform the following steps:

1. Ensure that the Nokia NFM-P server is configured to send data to the probe in UTF-8 format.

   The probe can support only multibyte characters if the incoming data has been configured in correct UTF-8 format. Consult your Nokia NFM-P server documentation for details about how to configure the server to send data in UTF-8 format.
2. Check that the probe server has UTF-8 support enabled and that the correct locale is set; for example, set the locale to Chinese.

   On Windows operating systems, use the following steps:

   a. Access the **Region and Language** section of the **Control Panel**.
   b. Select the **Formats** tab.
   c. Select **Format** > **Chinese (Simplified, PRC)**.
   d. Select the **Administrative** tab.
   e. Select **Change system locale**.
   f. Select **Current system locale** > **Chinese (Simplified, PRC)**.

g. Click **OK**.

h. Click **OK**.

On UNIX and Linux operating systems, set the system locale using the `LANG` and `LC_ALL` environment variables:

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

3. Restart the ObjectServer.
4. Configure the ObjectServer to enable the insertion of UTF-8 encoded data. See the *Netcool/OMNIbus Installation and Deployment Guide*.
5. If you are running the probe on a Windows operating system, you must use the `-utf8enabled` command-line option each time you start the probe.

Note: Netcool/OMNIbus can only receive multibyte characters from the Network Element (NE) custom properties using the `additionalText` field sent in the alarm. The custom text field also supports multibyte characters. However, this is an attribute of the alarm policy, which is not sent in the alarm. UTF-8 encoding is used for all fields in the Nokia NFM-P server, but multibyte character support only applies to these few selected attributes. This is because the NE does not support multibyte characters.

Running the probe

To start the probe on UNIX or Linux operating systems, use the following command: `$OMNIHOME/probes/nco-p-nokia-nfmp`

To start the probe on a Windows operating systems, use the following command: `%OMNIHOME%\probes\win32\nco-p-nokia-nfmp.bat`

Probe Event Enrichment Rules

The default enrichment rules are compatible with the Probe for Nokia NFM-P developed for IBM Tivoli Network Manager 4.1.1 or newer. Default enrichment rules which sets the LocalPriObj, NMosEventMap, LocalNodeAlias, RemoteNodeAlias for alarms to be enriched depending on the type of event map to be used.

A rules file specific to the Nokia NFM-P will be included in the Probe Extension Package guide. For details, see the Knowledge Center:

To use the rules files:

1. Download and install the Probe Extension package.
2. Edit the main probes rules files to uncomment the include statement to the event enrichment file. Update the file path of the enrichment file, for example include:

   `"$PROBE_EXT/eventenrichment/SAM5620/alcatel_5620_sam.enrichment.rules"`

   Users may customize this rules files to support more alarms for enrichment.

https://www.ibm.com/support/knowledgecenter/SSSHTQ_int/omnibus/probes/probe_ext_pack/wip/concept/probe_ext_pack_intro.html?lang=en

Data acquisition

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

The Probe for Nokia Network Functions Manager for Packet acquires current alarms and downtime alarms (if there is a loss of connection) by connecting to the HTTP port on the Nokia NFM-P server. The probe can also obtain alarms as they are generated by subscribing to the Java Messaging Service (JMS) running on the Nokia NFM-P server.

The probe obtains data from Nokia NFM-P as follows:

1. The probe connects to the machine on which the Nokia NFM-P server is running specified by the **Host** property.
2. If the **InitialResync** property is set to `true`, the probe connects to the **HTTPPort**, logs onto the NFM-P server using the values specified for the **NfmpServerUserName** and **NfmpServerPassword** properties, and sends an HTTP request to perform a resynchronization.

   The Nokia NFM-P server sends all currently active alarms.
3. The probe then connects to the **EJBPort** and subscribes to the JMS topic specified by the **JmsTopic** property to receive new events as they are generated.

   Note: You can specify filter criteria that the probe sends with the request using the **JmsFilter** property.

   The Nokia NFM-P server sends all events that meet the filter criteria to the probe.
4. The probe extracts event information using a SAX parser and sends it to the ObjectServer.
5. If the probe received a `TerminateClientSession` alarm from the Nokia NFM-P server, the probe ends the JMS session and shuts down.

Data acquisition is described in the following topics:

- "Connectivity and firewall considerations" on page 13
- "Resynchronization" on page 13
- "Using durable subscriptions" on page 14
- "Subscribing to JMS topics" on page 14
- "Subscribing to the 5620-SAM-topic-xml-filtered topic" on page 15
- "Filtering events" on page 16
- "JMS acknowlegement modes" on page 17
- "Nokia NFM-P server redundancy" on page 17
- "Peer-to-peer failover functionality" on page 18
- "Command line interface" on page 19

Connectivity and firewall considerations

To connect to the Nokia NFM-P server, the probe uses the **EJBPort** and **HTTPPort** properties.

The **EJBPort** property specifies the EJB port to which the probe connects. The default value for this property is 1099. The **HTTPPort** property specifies the HTTP port to which the probe connects. The default value for this property is 8080.

For connectivity within a firewall, Nokia NFM-P also uses the JMS port. The JMS service details are specified within the Nokia NFM-P server. The JMS service can only be altered from the Nokia NFM-P server configuration. The default value for the JMS port is 8093.

Resynchronization

On startup, you can instruct the probe to connect to an HTTP port of the Nokia NFM-P server, and send a SOAP/XML request to resynchronize all active alarms.

Using the **InitialResync** property, the probe can resynchronize alarms generated while the network elements managed by the NFM-P server are either `inService` (active) or `inMaintenance` (in the maintenance stage or inactive). The Nokia NFM-P server generates `inService` alarms when the associated network elements are active, and generates `inMaintenance` alarms when they are inactive.

To resynchronize only the active network elements and to receive only the `inService` alarms, set the **XMLretrieveUseInService** property to 1.

If you have specified a value for the **RecoveryFile** property, you must create the file manually. The server only sends alarms that have been generated since the timestamp indicated in the recovery file.

If you set the **ResynchInterval** property to a value greater than 1, the probe periodically resynchronizes with the Nokia NFM-P server. If you set the **ResyncInterval** property to a low value, you may experience performance issues if the probe has to resynchronize large amounts of active alarms each interval.

If you set the **Optimize** property to 1, the probe optimizes performance by not writing the resynchronized events received from the HTTP request and the JMS events to the log file.

Using durable subscriptions

The probe can make a durable or a non-durable subscription with a JMS topic.

In durable subscription to a topic, an active subscriber specified by the probe receives messages that were published on the topic while the subscriber was inactive. When the durable subscription is enabled, the JMS stores messages published on the specified topic while the subscriber is not active or disconnected from the JMS.

In a non-durable subscription to a topic, the subscriber only receives messages that are published on that topic while the subscriber is active. The JMS drops other messages that are sent while the subscriber is inactive.

Enabling durable subscriptions

You can specify the identifier of the subscriber in the **PersistentJmsId** property, and enable a durable subscription using the **Durable** property. When the probe reconnects with the same subscriber value, the JMS sends the stored events.

In durable subscription mode, if you stop the probe manually using **Ctrl - C** and the **StoreEvents** property is set to `0` the JMS session becomes inactive. However, when the probe is stopped unexpectedly the durable JMS session remains active (or if the **StoreEvents** property is set to 1). For example, if the probe is stopped due to a system crash, the JMS session remains active. This helps the probe receive alarms generated when it was inactive.

Note: Make sure that the same Jms `ClientId` value specified using the **PersistentJmsID** property is part of the **JmsFilter** property value.

Disabling durable subscriptions

To disable the durable subscription mode, set the value for the **Durable** property to `false`. If the value specified for this property is `false`, the probe makes a non-durable subscription with the JMS topic and receives active alarms.

Note: The JMS removes the stored messages once the probe receives new messages, when the stored messages expire, or when the durable subscription is deleted.

Subscribing to JMS topics

The probe can subscribe to a JMS topic and receive events in XML format.

The following table describes the topics to which the probe can subscribe using the **JmsTopic** property.

Table 5. JMS topics

JMS topic	Description
`5620-SAM-topic-xml-fault`	Subscribes to `fault` events in XML format.
`5620-SAM-topic-xml-file`	Subscribes to `findToFile` events in XML format.

Table 5. JMS topics (continued)

JMS topic	Description
`5620-SAM-topic-xml-general`	Subscribes to `general` events (such as object creation and deletion) in XML format.
`5620-SAM-topic-xml-stats`	Subscribes to events related to statistics collection.
`5620-SAM-topic-xml-filtered`	Subscribes to all events using advanced filtering to limit the number of JMS messages that are sent.

Subscribing to the 5620-SAM-topic-xml-filtered topic

The Probe for Nokia NFM-P can subscribe to the `5620-SAM-topic-xml-filtered` topic. This topic supports the advanced JMS message filter which allows you to base event message selection on the content of the event message body.

To subscribe to the `5620-SAM-topic-xml-filtered` topic, set the **`JmsTopic`** property to `5620-SAM-topic-xml-filtered.` The probe will receive events of type `EventVessel` and `XMLFilterChange`. You can limit the messages that the NFM-P sever sends to the probe by specifying an XML filter using the **`AdvancedFilterConfigFile`** property.

You can dynamically modify the XML message filter when the probe is running by issuing a `registerNotification` command from the CLI, or you can specify an alternative XML filter by issuing a `registerNotificationFile` command. For details of these commands, see "Command line interface" on page 19.

The default **`AdvancedFilterConfigFile`** contains the following XML code:

```
<filter-Set>
    <filter>
   <and>
    <equal class="jmsEvent" name="MTOSI_NTType" value="NT_ALARM" />
     <not>
       <equal class="jmsEvent" name="MTOSI_NTType"
            value="NT_ATTRIBUTE_VALUE_CHANGE" />
            </not>
    </and>
    </filter>
</filter-Set>
<extraTags>
    <tag name="ALA_category" />
    <tag name="eventName" />
    <tag name="MTOSI_osTime" />
    <tag name="MTOSI_objectType" />
    <tag name="MTOSI_perceivedSeverity" eventName="ObjectCreation" />
    <tag name="ALA_alarmType" eventName="ObjectCreation" />
    <tag name="ALA_clientId"  />
</extraTags>
```

When the probe subscribes to the `5620-SAM-topic-xml-filtered` topic, by default, the `event` vessel does not contain the JMS header properties of the individual events. You must use the **`AdvancedFilterConfigFile`** property to specify the JMS header properties (for example, MTOSI_osTime, eventName, and so on).

For information about how to configure complex filters for advanced filtering purposes, refer to the Nokia NFM-P documentation.

Note: The Probe for Nokia NFM-P only supports XML filters that have 90,000 characters or fewer. If the XML filter specified by the **`AdvancedFilterConfigFile`** property contains more than 90,000 characters, the probe writes the following warning message to the probe log file:

```
Probe only support AdvancedFilterConfigFile: ...
```

Filtering events

You can specify how the JMS filters the events sent to the probe by using the **`JmsFilter`** property. This property allows you to specify a filter that uses the JMS properties associated with the event.

You can filter events using any of the JMS properties. For example, the following filter is based on fault messages with a severity class of `Warning`:

```
ALA_category in (\'FAULT\') and MTOSI_perceivedSeverity in (\'Warning\')
```

By default, the JMS filters events sent to the probe using the following filter:

```
ALA_category not in (\'STATISTICS\', \'ACCOUNTING\')
```

Note: The JMS filter must contain `ALA_clientId in ('`*`persistent_jms_id`*`', '')` where *`persistent_jms_id`* is the value set for the **`PersistentJmsId`** property. If you change the value of the **`PersistentJmsId`** property, you must also change the value in the JMS filter accordingly.

The following table describes all JMS properties that are available for use in filters. For each property, the table lists the possible values that the property can take.

Table 6. JMS properties

Property	Values	Description

`MTOSI_NTType`	`ALA_OTHER`
	`NT_OBJECTCREATION`
	`NT_OBJECTDELETION`
	`NT_ATTRIBUTE_VALUE_ CHANGE`
	`NT_STATE_CHANGE`
	`NT_ALARM`
	`NT_HEARTBEAT`
	`ALA_ RELATIONSHIPCHANGE`
		Use this property to specify the type of notification that you want the filter to send to the probe.

`MTOSI_osTime`	Time in milliseconds	Use this property to specify the time when the event was created on the server. The filter sends this time of the event creation to the probe.
`MTOSI_objectName`	Object name	Use this property to specify the object name for the event that you want the filter to send to the probe.
`MTOSI_objectType`	Object type	Use this property to specify the object type for the event that you want the filter to send to the probe.

Table 6. JMS properties (continued)

Property	Values	Description

`ALA_category`	`SERVICE`
	`EQUIPMENT`
	`ACCOUNTING`
	`GENERAL`
	`FAULT`
	`STATISTICS`
	`DATABASE`
	`SOFTWARE`
		Use this property to specify the category of the event that you want the filter to send to the probe.

`ALA_allomorphic`	Allomorphic class for the object type	Use this property to specify the allomorphic class for the object type that you want the filter to send to the probe.
`ALA_topic`	JMS topic	Use this property to specify the JMS topic for the message that you want the filter to send to the probe.
`ALA_clientId`	Specific client ID	Use this property to specify the client ID that you want the filter to send to the probe.

JMS acknowlegement modes

Within the Nokia NFM-P server, the probe acts as a JMS client and supports either of the two acknowledgement modes: DUPS_OK_ACKNOWLEDGE or AUTO_ACKNOWLEDGE. You can set the **`JMSAcknowledgeMode`** property to a mode in which the probe should run as a JMS client.

If the JMS is running in DUPS_OK_ACKNOWLEDGE mode, the probe allows the JMS provider to send a message more than once to the same destination. This mode can be chosen when the performance and throughput of the probe decrease to avoid duplicate messages.

If the JMS is running in AUTO_ACKNOWLEDGE mode, the probe automatically acknowledges a received message, and waits for a broker acknowledgement of the message.

Nokia NFM-P server redundancy

Two Nokia NFM-P servers can run in a redundancy pair (that is, one runs as the primary server and the other as a backup server). This affects the way that you configure the probe.

If the primary NFM-P server is down while the probe is connected, then the probe will attempt to connect to the secondary NFM-P server which will take over the primary server role. The probe cannot connect to the secondary NFM-P server if the primary server is still operational.

This feature works with the **`RetryCount`** property and the optionally with the **`RetryInterval`** property.

If **`RetryCount`** is 0 (retry is disabled), the redundancy feature is also disabled.

The **`RetryInterval`** property should be set to the appropriate amount of time needed for the NFM-P server to complete the server activity switch.

When the **`EnableFailover`** property is set to true the probe is trying to setup a connection with the new primary NFM-P server and you should see the following messages.

ProbeWatch message:

```
Connecting new primary Nokia NSP NFM-P server.
```

Debug message:

```
Setting up connection to primary/secondary host.
```

When the probe is successfully connected to the primary NFM-P server, you will see the following ProbeWatch message:

```
Connected to the primary Nokia NSP NFM-P server.
```

These are the probe properties impacted by the NFM-P server redundancy:

- **Host** (The primary host in redundancy).
- **HTTPPort** (This is the primary HTTP port in redundancy).
- **EJBPort** (EJBPort is the primary EJB port in redundancy).
- **EnableFailover**
- **SecondaryHost**
- **SecondaryHTTPPort**
- **SecondaryEJBPort**

Note: If **EnableFailover** is set to true then the **RetryCount** must be set to a value greater than 0.

Peer-to-peer failover functionality

The Probe for Nokia Network Functions Manager for Packet supports failover configurations where two probes run simultaneously. One probe acts as the `master` probe, sending events to the ObjectServer; the other acts as the `slave` probe on standby. If the master probe fails, the slave probe activates.

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

Note: Peer-to-peer functionality does not work if you specify the same value for the **JmsFilter** property for both the master probe and the slave probe files. You must create two queues that both use the JMS filter that you want to use, and specify one queue in the master probe properties file and specify the other queue in the slave probe properties file. You must also specify different values for the **PersistentJmsID** property on the master and slave properties.

Example property file settings for peer-to-peer failover

The following settings show the peer-to-peer settings from the properties file of an example master probe:

```
PidFile    : "master_pid_file"'
Server        :    "NCOMS"
RulesFile   :    "master_rules_file"
MessageLog  :    "master_log_file"
PeerHost    :    "slave_hostname"
PeerPort    :     5555  # [communication port between master and slave probes]
Mode        :    "master"
JmsFilter: ALA_clientId in ('netcool@master@127.0.0.1', '') and ALA_category not in
('STATISTICS', 'ACCOUNTING')
PersistentJmsId : netcool@master@127.0.0.1
```

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

```
PidFile   : "slave_pid_file"'
Server      :    "NCOMS"
RulesFile   :    "slave_rules_file"
```

```
MessageLog  :    "slave_log_file"
PeerHost    :    "master_hostname"
PeerPort    :     5555 # [communication port between master and slave probes]
Mode        :    "slave"
JmsFilter: ALA_clientId in ('netcool@slave@127.0.0.1', '') and ALA_category not in
('STATISTICS', 'ACCOUNTING')
PersistentJmsId : netcool@slave@127.0.0.1
```

Note: The properties files also contain all other properties required to configure the probe.

Command line interface

The probe is supplied with a command line interface (CLI) that allows you to manage the probe while it is running. For IBM Tivoli Netcool/OMNIbus V7.4 and later, use the HTTP/HTTPS command interface.

HTTP/HTTPS command interface

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

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

The following sections show:

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

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

Configuring the command interface

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

NHttpd.EnableHTTP: Set this property to `True`.

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

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

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

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

Format of the nco_http command line

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

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

Where:

- *probeuri* is the URI of the probe.
- *probeport* is the port that the probe uses to listen for HTTP/HTTPS commands. Specify the same value as that set for the **NHttp.ListeningPort**.

- *command-name* is the name of the command to send to the probe. The following command names are available:

  help
  name
  registerNotification
  registerNotificationFile
  resync
  shutdownprobe
  version

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

Probe commands

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

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

help

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

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

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

The following command returns help information:

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

name

Use the **name** command to display the name of the probe.

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

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

The following command displays the name of the probe:

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

registerNotification

Use the **registerNotification** command to register the advanced JMS filter XML file specified by the **AdvancedFilterConfigFile** property.

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

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

The following command registers the advanced JMS filter XML file specified by the **AdvancedFilterConfigFile** property.:

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

registerNotificationFile

Use the **registerNotificationFile** command to register an alternative JMS filter XML file.

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

```
-data '{"command":"registerNotificationFile", "params":
[{"XML_config_file_path":"alternative_jms_filter"}]}'
```

Where *alternative_jms_filter* is the path to the alternative JMS filter XML file that you want to register.

The following command registers the file /opt/IBM/tivoli/netcool/omnibus/var/registerNotifcaton.txt as the alternative JMS filter XML file:

```
$OMNIHOME/bin/nco_http -uri http://localhost:8080/probes/nokia_nfmp -datatype
application/JSON -method POST -data '{"command":"registerNotificationFile",
"params":[{"XML_config_file_path":"/opt/IBM/tivoli/netcool/omnibus/var/
registerNotifcaton.txt"}]}'
```

resync

Use the **resync** command to perform a resynchronization using the parameters specified by the following properties in the properties file:

- **XMLretreiveUseInService**
- **NfmpServerUserName**
- **NfmpServerPassword**

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

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

The following command performs a resynchronization:

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

shutdownprobe

Use the **shutdownprobe** command to shut down the probe.

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

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

The following command shuts down the probe:

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

version

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

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

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

The following command returns version information:

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

Messages in the log file

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

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

```
Information: I-UNK-104-002: {"response":["Resync successful"],"status":"200"}
```

Storing commands in the nco_http properties file

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

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

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

Data
DataType
Method
URI

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

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

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

Properties and command line options

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

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

Table 7. Properties and command line options

Property name	Command line option	Description
`AdvancedFilterConfigFile` *string*	`-advancedfilter configfile` *string*	Use this property to specify the path of an advanced JMS filter XML file. The Nokia NFM-P server uses this file to determine which events to send to the probe when the **`JmsTopic`** property is set to `5620-SAM-topic-xml-filtered`. For details about how to dynamically change the advanced JMS filter while the probes is running, see "Subscribing to the 5620-SAM-topic-xml-filtered topic" on page 15. On UNIX and Linux operating systems, the default is `$OMNIHOME/var/registerNotification.txt`. On Windows operating systems, the default is `%OMNIHOME%\\var\\registerNotification.txt`.
`CertificateStore` *string*	`-certificatestore` *string*	Use this property to specify the path of the certificate store file. The default is `""`.
`CertificateStorePassword` *string*	`-certificatestore password` *string*	Use this property to specify the password required to access the certificate store file. The default is `""`.
`CertificateStoreType` *string*	`-certificatestore type` *string*	Use this property to specify the certificate type obtained from the certificate store. The default is `JKS`.
`Durable` *string*	`-durable` *string*	Use this property to specify that the probe makes a durable subscription to the JMS. `false`: JMS subscription is not durable. `true`: JMS subscription is durable. The default is `true`.
`EJBPort` *integer*	`-ejbport` *integer*	Use this property to specify the EJB port to which the probe connects. The default is `1099`.
`EnableFailover` *string*	`-noenablefailover` (This is equivalent to **`EnableFailover`** with a value of false.) `-enablefailover` (This is equivalent to **`EnableFailover`** with a value of true.)	Use this property to specify an instance of probe to support redundant NFM servers. The default is `false`. If **`EnableFailover`** is set to true then the **`RetryCount`** must be set to a value greater than `0`.

Table 7. Properties and command line options (continued)

Property name	Command line option	Description
Host *string*	-host *string*	Use this property to specify the IP address of the host machine on which the Nokia NFM-P server is running. The default is `localhost`.
HTTPPort *integer*	-httpport *integer*	Use this property to specify the HTTP port to which the probe connects. The default is `8080`.
JMSAcknowledgeMode *string*	-jmsacknowledge mode *string*	Use this property to specify the acknowledgment mode in which the probe runs. This property takes the following values: `AUTO_ACKNOWLEDGE`: The probe runs in automatic acknowledgement (AUTO_ACKNOWLEDGE) mode. `DUPS_OK_ACKNOWLEDGE`: The probe runs in deduplication allowed (DUPS_OK_ACKNOWLEDGE) mode. The default is `DUPS_OK_ACKNOWLEDGE`
JmsFilter *string*	-jmsfilter *string*	Use this property to specify the filter the probe uses to limit the events received from Nokia NFM-P. For details of the format in which to specify this filter, see "Filtering events" on page 16. The default value is `ALA_clientId in ('netcool@127.0.0.1', '') and ALA_category not in (\'STATISTICS\', \'ACCOUNTING\')`.
JmsTopic *string*	-jmstopic *string*	Use this property to specify the XML topic that the probe uses to subscribe to events within the JMS. For details about valid values for this property, see Table 5 on page 14. The default is `5620-SAM-topic-xml`.
NfmpServerPassword *string*	-nfmpserverpasswor d *string*	Use this property to specify the password required with the **NfmpServerUserName** property to log in to the JMS. The default is `""`.
NfmpServerUserName *string*	-nfmpserverusernam e *string*	Use this property to specify the user name with which the probe logs in to the JMS. The default is `""`.

Table 7. Properties and command line options (continued)

Property name	**Command line option**	**Description**
`Optimize` *integer*	`-optimize` *integer*	Use this property to specify whether the probe optimizes performance: `0`: The probe does not optimize performance. `1`: The probe optimizes performance by writing neither events received from the HTTP request nor JMS events to the log file. The default is `1`.
`ParseSaxDebug` *integer*	`-parsesaxdebug` *integer*	Use this property to specify whether the probe outputs extra debug information from the SAX Parser. The default is `0`. **Note:** Use this option for debug purposes only.
`PersistentJmsId` *string*	`-persistentjmsid` *string*	Use this property to specify the ID of the JMS subscription. The default is `netcool@127.0.0.1`.
`RecoveryFile` *string*	`-recoveryfile` *string*	Use this property to specify the name of the recovery file in which the probe stores the timestamp of the last `keepAlive` event read before disconnecting from the Nokia NFM-P server. The probe uses this when reconnecting to the server. On UNIX and Linux operating systems, the default is `$OMNIHOME/var/NFMPRecovery`. On Windows operating systems, the default is `%OMNIHOME%\\var\\NFMPRecovery`. This recovery file does not exist by default. You must create this file manually when this property is specified for the first time. **Note:** You should backup or remove the recovery file when switching to a new machine or upgrading to a newer version of the probe. Failure to do so may result in successive resynchronizations but with missing events.
`ResyncBatchSize` *integer*	`-resyncbatchsize` *integer*	Use this property to specify the maximum number of alarms that the probe retrieves in each batch of resynchronization alarms. The default is `100`.

Table 7. Properties and command line options (continued)

Property name	Command line option	Description
SecondaryEJBPort *integer*	-secejbport *integer*	Use this property to specify the EJB port of secondary host machine to which the probe connects. The default is 1099.
SecondaryHost *string*	-sechost *string*	Use this property to specify the IP address of the secondary host machine on which the Nokia NFM-P server is running.
SecondaryHTTPPort *integer*	-sechttpport *integer*	Use this property to specify the HTTP port of the secondary host machine to which the probe connects. The default is 8080.
StoreEvents *integer*	-storeevents *integer*	Use this property to specify whether Nokia NFM-P continues to store events after the probe has been stopped by a CTRL-C command. This property takes the following values: 0: Nokia NFM-P does not store events after a CTRL-C stop. 1: Nokia NFM-P stores events after a CTRL-C stop. The default is 1.
TrustStore *string*	-truststore *string*	Use this property to specify the file path of the truststore file. The default is "".
TrustStorePassword *string*	-truststorepassword *string*	Use this property to specify the password required to access the truststore file containing the trusted certificates. The default is "".

Table 7. Properties and command line options (continued)

Property name	Command line option	Description
UseSSL *string*	`-usessl` *string*	Use this property to enable SSL authentication when connecting to the Nokia NFM-P server. This property takes the following values: `false`: SSL is not used when connecting to the Nokia NFM-P server. `true`: SSL is used when connecting to the Nokia NFM-P server. The default is `false`. **Note:** If **UseSSL** is set to true, the **HTTPPort** property must be configured to use the Nokia NFM-P HTTPS port. The default Nokia NFM-P port is 8443. You must also configure the following properties for the Probe for Nokia Network Functions Manager for Packet: • **CertificateStoreType** • **CertificateStore** • **CertificateStorePassword** • **TrustStore** • **TrustStorePassword**
XMLFile *string*	`-xmlfile` *string*	Use this property to specify the XML file that the probe reads to check the SAX parser. The default is `""`. **Note:** Use this option for debug purposes only.
XMLFileRead *integer*	`-xmlread` *integer*	Use this property to specify whether the probe reads an XML file to check the SAX parser. This property takes the following values: `0`: The probe does not read an XML file. `1`: The probe reads the XML file specified by the **XMLFile** property. The default is `0`. **Note:** Use this option for debug purposes only.

Table 7. Properties and command line options (continued)

Property name	Command line option	Description
XMLretrieveUseInService *integer*	-xmlretrieveuseinservice *integer*	Use this property to specify whether the probe receives alarms from the nodes that are in either inService or inMaintenance mode. This property takes the following values: 0: The probe receives alarms from all the nodes. 1: The probe receives alarms only from the nodes that are in the inService mode. The default is 0.

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

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

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

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

Table 8. 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.
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.

Table 8. Properties and command line options (continued)

Property name	Command line option	Description
Inactivity *integer*	`-inactivity` *integer*	Use this property to specify the length of time (in seconds) that the probe allows the port to receive no incoming data before disconnecting. The default is `0` (which instructs the probe to not disconnect during periods of inactivity).
InactivityAction *string*	`-inactivityaction` *string*	Use this property to specify the action the probe takes when the inactivity timeout is reached: `SHUTDOWN`: The probe sends a ProbeWatch message to notify the user and then shuts down. `CONTINUE`: The probe sends a ProbeWatch message to notify the user, but does not shut down. The default is `SHUTDOWN`.
InitialResync *string*	`-initialresync` *string*	Use this property to specify whether the probe performs resynchronization on startup. This property takes the following values: `false`: The probe does not request resynchronization on startup. `true`: The probe requests resynchronization on startup. For most probes, the default value for this property is `false`. If you are running the JDBC Probe, the default value for the **InitialResync** property is `true`. This is because the JDBC Probe only acquires data using the resynchronization process.
MaxEventQueueSize *integer*	`-maxeventqueuesize` *integer*	Use this property to specify the maximum number of events that can be queued between the non native process and the ObjectServer. The default is `0`. **Note:** You can increase this number to increase the event throughput when a large number of events is generated.

Table 8. Properties and command line options (continued)

Property name	Command line option	Description
`ResyncInterval` *integer*	`-resyncinterval` *integer*	Use this property to specify the interval (in seconds) at which the probe makes successive resynchronization requests. For most probes, the default value for this property is `0` (which instructs the probe to not make successive resynchronization requests). If you are running the JDBC Probe, the default value for the **`ResyncInterval`** property is `60`. This is because the JDBC Probe only acquires data using the resynchronization process.
`RetryCount` *integer*	`-retrycount` *integer*	Use this property to specify how many times the probe attempts to retry a connection before shutting down. The default is `0` (which instructs the probe to not retry the connection).
`RetryInterval` *integer*	`-retryinterval` *integer*	Use this property to specify the length of time (in seconds) that the probe waits between successive connection attempts to the target system. The default is `0` (which instructs the probe to use an exponentially increasing period between successive connection attempts, for example, the probe will wait for 1 second, then 2 seconds, then 4 seconds, and so forth).
`RotateEndpoint` *string*	`-rotateendpoint` *string*	Use this property to specify whether the probe attempts to connect to another endpoint if the connection to the first endpoint fails. This property takes the following values: `false`: The probe does not attempt to connect to another endpoint if the connection to the first endpoint fails. `true`: The probe attempts to connect to another endpoint if the connection to the first endpoint fails. The default is `false`.

Elements

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

This section describes the elements that the Probe for Nokia Network Functions Manager for Packet JMS generates.

Common elements

The probe generates a common set of elements for each event that it receives.

The following table describes the elements that are generated for all events.

Table 9. Common elements

Element name	**Element description**
`$ALA_allomorphic`	This element contains the allomorphic class of the object.
`$ALA_category`	This element shows the category of the message, for example, FAULT.
`$ALA_clientId`	This element contains the client identifier for the message.
`$correlationId`	This element contains the correlation identifier for the related messages.
`$eventName`	This element shows the type of event, for example, an attribute value change.
`$MTOSI_NTType`	This element shows the notification type of the message, for example, NT_ALARM.
`$MTOSI_objectName`	This element contains the object full name associated with the event.
`$MTOSI_objectType`	This element indicates the object type for the message, for example, fm.AlarmObject.
`$MTOSI_osTime`	This element shows the time that the message was created.

Alarm event elements

The probe generates a standard set of elements for each alarm event that it receives.

The following table describes the elements that are generated for alarm event elements.

Table 10. Alarm event elements

Element name	**Element description**
`$ALA_alarmType`	This element indicates the alarm type.
`$ALA_OLC`	This element indicates the OLC state of the object associated to the event. A value of zero (0) indicates no state for the object.

Table 10. Alarm event elements (continued)

Element name	Element description
`$ALA_application`	This element shows the tag associated with the application.
`$ALA_span`	This element contains the span of control ID list.
`$ALA_isCorr`	This element shows the `isCorrelated` flag. A value of true indicates that the alarm is correlated at the creation time to another alarm.
`$MTOSI_aliasNameList`	This element indicates the alias for the alarm name.
`$MTOSI_perceivedSeverity`	This element indicates the severity of the alarm.
`$MTOSI_probableCause`	This element identifies the probable cause of the alarm.
`$MTOSI_serviceAffecting`	This element identifies whether the alarm is service affecting.

Attribute value change event elements

An attribute value change alarm indicates changes to a particular problem alarm that the probe received earlier. Based on the indicated changes, the probe generates a standard set of elements for each attribute value change event that it receives.

The probe performs the following if the attribute value change alarm indicates that the severity field is updated with a value other than cleared in `Attribute4`:

- Deduplicates the problem alarm.
- Updates severity of the problem alarm with the latest value.
- Increments the count of the problem alarm by 1.

The probe performs the following if the attribute value change alarm indicates that the severity field is updated with a cleared value in `Attribute4`:

- Makes the attribute value change alarm as a new resolution alarm in the event list. This action is performed as per the Netcool/OMNIbus generic clear standards.
- Clears both the problem alarm and the resolution alarm.

It is possible that the probe may receive another attribute value change alarm before the problem alarm was cleared. If this new attribute value change alarm indicates that the severity field is updated with a value other than cleared in `Attribute4`, then the probe performs the following:

- Deduplicates the problem alarm.
- Updates severity of the problem alarm with the latest value.
- Increments the count of the problem alarm by 1.

It is The probe performs the following if the attribute value change alarm indicates that attributes other than the severity field are updated in `Attribute4`:

- Deduplicates the problem alarm.
- Keeps the severity field with its current value.
- Increments the count of the problem alarm by 1.

The following table describes the elements that are generated for attribute value change events.

Table 11. Attribute value change event elements

Element name	Element description
`$avc_`*n*	This element shows the attribute changed and its associated value.
`$objectFullName`	This element contains the name of the object whose attributes changed.

State change event elements

The probe generates a standard set of elements for each state change event that it receives.

The following table describes the elements that are generated for state change event elements.

Table 12. State change event elements

Element name	Element description
`$maxAlarmCount`	This element shows the maximum alarm count.
`$state`	This element indicates the state that changed.
`$sysPrimaryIp`	This element shows the primary IP address of the system.
`$sysStandbyIp`	This element shows the standby IP address of the system.
`$sysStartTime`	This element shows the time that the system started up.

Alarm status change event elements

The probe generates a standard set of elements for each alarm status change event that it receives.

The following table describes the elements that are generated for alarm status change event elements.

Table 13. Alarm status change event elements

Element name	Element description
`$Attribute`*n*	This element contains the attribute changed and its associated value.
`$objectFullName`	This element shows the name of the object whose alarm status has changed.
`$olcState`	This element indicates whether the alarm is generated when the node is in one of these modes: `inService` or `inMaintenance`.
`$spanObjectPointer`	This element identifies the span of object pointer.
`$correlatingAlarm`	This element indicates whether the alarm has a correlating alarm.
`$isImplicitlyCleared`	This element indicates whether the alarm is implicitly cleared.

Table 13. Alarm status change event elements (continued)

Element name	Element description
`$numberOfCorrelatedAlarms`	This element shows the number of alarms correlated to the alarm.

Object deletion event elements

The probe generates a single element for each database proxy state change event that it receives.

The following table describes the element that is generated for object deletion event elements.

Table 14. Object deletion event elements

Element name	Element description
`$objectFullName`	This element shows the name of the object deleted.

Relationship change event elements

The probe generates a standard set of elements for each relationship change event that it receives.

The following table describes the elements that are generated for relationship change event elements.

Table 15. Relation change event elements

Element name	Element description
`$changeType`	This element shows the type of change.
`$fromObjectClass`	This element contains the class of the original object.
`$fromObjectName`	This element identifies the name of the original object.
`$objectFullName`	This element identifies the name of the object with a changed relationship.
`$toObjectClass`	This element contains the class of the new object.
`$toObjectName`	This element shows the name of the new object.

Elements generated for object creation events

The probe generates a standard set of elements for each object creation event that it receives.

The following table describes the elements generated when alarm information objects are created.

Table 16. Elements for object creation events

Element name	Element description
`$acknowledgedBy`	This element contains the name of the operator that acknowledged the alarm.
`$additionalText`	This element contains the additional descriptive information about the alarm.

Table 16. Elements for object creation events (continued)

Element name	**Element description**
`$affectedObjectClassIndex`	This element indicates the identifier of the managed object class.
`$affectedObjectClassName`	This element contains the name of the managed object class to which the object with the alarm belongs.
`$affectedObjectDisplayedName`	This element contains the text version name of the alarm's affected object.
`$affectedObjectFullName`	This element contains the fully distinguished name of the object against which the alarm was raised.
`$affectedObjectInstanceIndex`	This element indicates the identifier of the managed object instance.
`$alarmClassTag`	This element shows the object class against which the alarm was raised.
`$alarmName`	This element contains the alarm name.
`$alarmType`	This element shows the vendor-specific and X.733 standards for the type of the alarm.
`$firstTimeDetected`	This element shows the date and time the alarm was first raised.
`$highestSeverity`	This element identifies the highest severity value of the alarm since it was generated.
`$isAcknowledged`	This element indicates whether the alarm is currently acknowledged by an operator.
`$isServiceAffecting`	This element indicates whether an alarm has affected a customer's service.
`$lastTimeAcknowledged`	This element shows the date and time at which the alarm was acknowledged.
`$lastTimeCleared`	This element shows the date and time at which the alarm was cleared.
`$lastTimeDeEscalated`	This element shows the date and time at which the alarm was de-escalated.
`$lastTimeDemoted`	This element shows the date and time at which the alarm was demoted.
`$lastTimeDetected`	This element shows the date and time the alarm was most recently observed.

Table 16. Elements for object creation events (continued)

Element name	**Element description**
`$lastTimeEscalated`	This element shows the date and time at which the alarm was escalated.
`$lastTimePromoted`	This element shows the date and time at which the alarm was promoted.
`$lastTimeSeverityChanged`	This element shows the date and time at which the alarm was cleared.
`$nodeId`	This element contains the IP address of the router that issued the alarm and contains the affected object.
`$nodeName`	This element contains the text version name of the router that issued the alarm and contains the affected object.
`$numberOfOccurrences`	This element indicates the number of times the alarm has been raised.
`$numberOfOccurrencesSinceAck`	This element indicates the number of times the alarm has been raised since it was acknowledged.
`$numberOfOccurrencesSinceClear`	This element indicates the number of times the alarm has been raised since it was cleared.
`$operatorAssignedUrgency`	This element indicates the urgency setting of the alarm as determined by the user-configured value set for the alarm. The possible values are: 0: Unspecified 1: Indeterminate 2: Minor 3: Major 4: Critical
`$originalSeverity`	This element shows the severity of the alarm when it was first raised.
`$previousSeverity`	This element shows the severity value of the alarm previous to the current value.
`$probableCause`	This element shows the vendor-specific and X.733 standards for the probable cause of the alarm.
`$relatedObject`	This element contains the list of fully distinguished names that point to other objects that are also affected by a particular alarm.

Table 16. Elements for object creation events (continued)

Element name	Element description
`$specificProblem`	This element indicates the specific problem that the alarm is reporting. It is generated when the node reports vendor specific information in the `specificProblem` field. The probe populates this element by referring to the `nokia_nfmp.lookup` table supplied with the probe. For details about configuring the lookup table, see "Configuring lookup tables" on page 6.
`$severity`	This element shows the vendor-specific, TMN, and X.733 standards for severity of the alarm. The possible values are: 0: Unspecified 1: Cleared 2: Indeterminate 3: Info 4: Condition 5: Warning 6: Minor 7: Major 8: Critical
`$urgencyAssignedBy`	This element contains the name of the operator that last modified the urgency of the alarm.
`$userText`	This element indicates custom user text of the alarm.
`$wasAcknowledged`	This element indicates whether the alarm has ever been acknowledged by an operator.

Customizing the rules file to populate the $specificProblem element

The `$specificProblem` element is available for NFM-P for certain types of alarms, for example NobeB alarms. To populate the `$specificProblem` element you need to customize the `nokia_nfmp.rules` file.

The probe rules file contains the following lines that define the `specificProblem` element:

```
 #specificProblem is only available starting SAM v12 and above; and applicable
for certain alarm type only (eg: NodeB alarm)
    if (exists($specificProblem))
    {
        $SpecificProblem_T = lookup($specificProblem, SpecificProblem_t)
        ## Remove hash below to map specific problem with object server field
        ## @X733SpecificProb = $SpecificProblem_T
    }
```

To make the `$specificProblem` element available, uncomment the following line by removing the hash symbols:

```
@X733SpecificProb = $SpecificProblem_T
```

This will map `$SpecificProblem` to the `@X733SpecificProb` field.

Alternatively, you can create a new field in the ObjectServer using Netcool Administrator and map `$SpecificProblem_T` with a new field (for example `@NewField`).

```
@NewField = $SpecificProblem_T
```

Client session event elements

The probe generates a single element for each client session event that it receives.

The following table describes the element that is generated for client session event elements.

Table 17. Client session event elements

Element name	Element description
`$clientId`	This element shows the identifier of the client in this session.

File available event elements

The probe generates a standard set of elements for file available events that it receives.

The following table describes the elements that are generated for file available event elements.

Table 18. File available event elements

Element name	Element description
`$fileName`	This element contains the name of the file required.
`$requestId`	This element shows the identifier of the request.

Database activity event elements

The probe generates a single element for each database activity event that it receives.

The following table describes the element that is generated for database activity event elements.

Table 19. Database activity event elements

Element name	Element description
`$state`	This element shows the state associated with the database activity.

Database connection state change event elements

The probe generates a single element for each database connection state change event that it receives.

The following table describes the element that is generated for database connection state change event elements.

Table 20. Database connection state change event elements

Element name	Element description
`$state`	This element indicates the state of the database connection.

Database error event elements

The probe generates a standard set of elements for each database error event that it receives.

The following table describes the elements that are generated for database error events.

Table 21. Database error event elements

Element name	Element description
`$error`	This element shows the description of the error.
`$state`	This element identifies the database error state.

Database proxy state change event elements

The probe generates a single element for each database proxy state change event that it receives.

The following table describes the element that is generated for database proxy state change events.

Table 22. Database proxy state change event elements

Element name	Element description
`$state`	This element indicates the state of the database proxy.

Error messages

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

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

Table 23. Error messages

Error	Description	Action
`Failed to get properties`	The probe could not access the properties file.	Check the location of and permissions set for the properties file.
`Failed to make HTTP connection to` *`host`*	The probe was unable to connect to the port specified by the **`HTTPPort`** property.	Check that you have specified the correct port in the properties file.
`JNDI API lookup failed`	The probe failed to create a `topicConnectionFactory` object with which to connect to the JMS.	Contact IBM Software Support.
`Please check whether you have configured incorrect CertificateStore/ TrustStore password.`	The JNDI API lookup failed due to an incorrect certificate or truststore password.	Check that you have configured the **`CertificateStore`** and **`TrustStore`** passwords correctly.

Table 23. Error messages (continued)

Error	Description	Action
`Please check whether you have configured correct TrustStore & TrustStorePassword in probe.`	The JNDI API lookup failed due to missing truststore configuration.	Check that you have configured the **`TrustStore`** and **`TrustStorePassword`** properties correctly.
`It looks like the JMS is configured to use SSL, and the probe is not properly configured to use it. Please double-check the probe configuration.`	The NFM-P JMS server has been configured to use SSL, but the probe is not properly configured to use it.	Check the probe configuration: Make sure the **`UseSSL`** property is set to `true` and that the related SSL properties have been configured correctly.
`Please ensure you have configure CertificateStore & CertificateStorePassword correctly in probe.`	The JNDI API lookup failed due to missing certificate store configuration.	Check that you have configured the **`CertificateStore`** and **`CertificateStorePassword`** properties correctly.
`Please check whether you have Host & EJBPort configured correctly in probe.`	The JNDI API lookup failed due to a `NoRouteToHost` exception.	Check that you have configured the **`TrustStore`** and **`TrustStorePassword`** properties 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 messages that the probe generates. For information about generic Netcool/OMNIbus ProbeWatch messages, see the *IBM Tivoli Netcool/OMNIbus Probe and Gateway Guide*.

Table 24. ProbeWatch messages

ProbeWatch message	Description	Triggers/causes
`Could not find JNDI context topic` *`topic_name`*	There was an error when the probe attempted to connect to the NFM-P server.	The JMS did not start properly on the NFM-P server.
`Failed to establish HTTP connection for resync`	The probe has failed to establish an HTTP connection to the NFM-P server.	The probe failed to connect to the port specified by the **`HTTPPort`** property.
`Going Down`	The probe is shutting down.	The probe is shutting down after performing the shutdown routine.

Table 24. ProbeWatch messages (continued)

ProbeWatch message	**Description**	**Triggers/causes**
`JNDI API lookup failed`	There was an error when the probe attempted to connect to the NFM-P server.	The JMS did not start properly on the NFM-P server.
`Running ...`	The probe is running normally.	The probe has just been started up.
`Unsupported Java class version error`	An unsupported Java class version exception occurred while the probe was initializing the JMS connection to the EMS.	The server is running a version of Java other than 1.8.
`xmlFilterChangeEvent received, advanced filtering registration is successful`	The probe successfully completed an advanced filtering registration.	The probe received an `xmlFilterChangeEvent` from the NFM-P server.
`Failed to register advanced filtering`	This indicates that the advanced filtering failed to register.	The NFM-P server returned an exception status of 1. Check the advanced XML filter specified by the **`AdvancedFilterConfigFile`** property and retry running the probe.
`Some classes are missing from the CLASSPATH. Exiting.`	The probe could not find the necessary class during connection.	Check that you have copied the necessary JAR files from the NFM-P server into the Netcool/OMNIbus environment.
`Unable to parse event due to IOException`	The probe could not parse events due to an `IO` exception. This may cause event loss.	Contact IBM support for assistance.
`Unable to parse event due to SAX Parser exception`	The probe could not parse events due to a `SaxParser` exception. This may cause event loss.	Contact IBM support for assistance.
`START SYNCHRONIZATION`	A resynchronization operation started.	The probe started receiving active alarms from EMS.
`STOP SYNCHRONIZATION`	A resynchronization operation completed.	The probe stopped receiving active alarms from EMS.
`Connecting new primary NFM-P server`	This ProbeWatch message is used to notify user about the probe is setting up connection to new primary NFM-P server after detected NFM-P activity switch in redundant NFM-P system.	
`Connected to primary NFM-P server`	The probe is connected to primary NFM-P server.	

Troubleshooting

Various issues arise as users work with the probe. Troubleshooting information is provided to help you diagnose and resolve such issues.

Probe is missing some events and how to resolve it?

Part of the value set for the **`JmsFilter`** property includes the value set for the **`PersistentJmsId`** property. So, the probe will not work properly if the **`JmsFilter`** property is not updated after changing the value of the **`PersistentJmsId`** property. When the probe encounters this issue, the following debug message appears in the logfile:

```
Debug: NFM-P: Sleeping until receiving the NFM-P server's data?
```

To resolve this problem, set the JMS filter and the JMS ID to the same IP address or host name, and not to a combination of local host and IP or host name.

Missing SystemInfoEvent message upon initial JMS connection

The NFM-P server is not currently sending the expected `SystemInfoEvent` message when the probe makes a JMS connection. If you have enabled inactivity processing (that is, set the **`Inactivity`** property in the probe properties file to a value greater than `0`), the probe will shut down when it does not receive the `SystemInfoEvent` message within the number of seconds defined in the **`Inactivity`** property during the check subscription process.

The workaround for this issue is to disable inactivity processing by setting the **`Inactivity`** property in the `nokia_nfmp.props` file to `0`.

JmsMissedEvent messages are not being sent when events have been lost

The NFM-P server is not currently sending the expected `JmsMissedEvents` messages when events have been lost. This means that the probe is not able to force a resynchronization in order to recover the missed events.

There is no workaround for this issue because since the `JmsMissedEvents` message is not sent by the NFM-P server, the probe has no way of knowing to send a resynchronization request. If you notice that there is event loss, you should request a resynchronization manually through the command port or HTTP interface to recover the lost events.

Known Issues

At the time of release, a number of known issues were reported that you should be aware of when running the probe.

Modifying the default Netcool/OMNIbus deduplication triggers for the Probe for Nokia NFM-P

The ObjectServer Summary field may change after receiving an AttributeValueChange event, but it should not.

To resolve this issue, run the `modify_dedup_trigger.sql` SQL script provided in the probe's installation package. This script modifies the default Netcool/OMNIbus deduplication triggers for the Probe for Nokia NFM-P to prevent the `@Summary` field from being updated when the ObjectServer receives an AttributeValueChange event.

The script is located in the following directory:

`$OMNIHOME/probes/arch`

Run the script using one of the following commands:

On UNIX and Linux operating systems use:

```
$OMNIHOME/bin/nco_sql -server objectserver_name -user username -password
password < path_to_file/modifyt_dedup.trigger.sql
```

On Windows operating systems use:

```
%NCHOME%\bin\redist\isql.exe -S objectserver_name -U username -P password -i
path_to_file\modify_dedup_trigger.sql
```

Where :

objectserver_name is the name of the ObjectServer against which you want to run the SQL file.

username is the username with which you log in to the ObjectServer

password is the password associated with the username you specified.

path-to_file is the path to the SQL file.

Disabling Automatic Alarm Deletion under Alarm Settings

If the user selects Disable Automatic Alarm Deletion under Alarm Settings and changes the severity of an event to "cleared" on NFM-P client GUI then in the event list the severity of the event will be changed to indeterminate.

This is to avoid the event from being deleted by delete_clears trigger. However, if the cleared event is deleted on NFM-P client GUI then the event will be resolved and removed from event list.

Using cross-launch with Internet Explorer

The probe is unable to cross-launch using the Internet Explorer browser from the **Active Event List** because Internet Explorer is no longer supported by NSP 21.6. Use the **Event Viewer** for cross-launch instead.

Integrating the probe with Nokia NFM-P 22.6

There is a connectivity issue with Nokia NFM-P whereby the probe can miss the `SystemInfoEvent` when resubscribing or reconnecting to the previous durable session. The probe durable session that showed in the Nokia NFM-P web client messaging connection may disappear when the probe is still connected. This issue is related to having a dot in the probe client ID name, for example:. `netcooljen@127.0.0.1`.

Workaround

Use a probe `clientId` without dot: in the name:

```
StoreEvents: 1
Durable: 'true'
PersistentJmsId                  : 'netcoolcert@hostname'
JmsFilter                        : 'ALA_clientId in (\'netcoolcert@hostname\', \'\') and
ALA_category not in (\'STATISTICS\', \'ACCOUNTING\')'
```

Appendix A. Notices and Trademarks

This appendix contains the following sections:

- Notices
- Trademarks

Notices

Trademarks

SC27-8758-06