Netcool/OMNIbus Probe for Network Node Manager-i
4.0

Reference Guide
June 26, 2020

SC27-6551-03

Note

Before using this information and the product it supports, read the information in Appendix A, “Notices and Trademarks,” on page 25.

Edition notice

This edition (SC27-6551-03) applies to version 4.0 of the Probe for NNMi and to all subsequent releases and modifications until otherwise indicated in new editions.

This edition replaces SC27-6551-02.

Contents

About this guide

The following sections contain important information about using this guide.

Document control page

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

The IBM Tivoli Netcool/OMNIbus Probe for Network Node Manager-i documentation is provided in softcopy format only. To obtain the most recent version, visit the IBM® Tivoli® Knowledge Center:

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

Table 1. Document modification history

Document version	Publication date	Comments
SC27-6551-00	November 7, 2014	First IBM publication. The probe supports NNMi 9.2.
SC27-6551-01	March 10, 2016	"Summary" on page 1. Support extended to include NNMi 10.0 and NNMi 10.10. Descriptions for the following properties were added to "Properties and command line options provided by the Java Probe Integration Library (probe-sdk-java) version 8.0" on page 18. • **EnableSSL** • **EnableSSLDebug** • **KeyStore** • **KeyStorePassword** • **KeyStoreType** • **SecurityProtocol** • **SubscriptionFilter** • **TrustStore** • **TrustStorePassword** • **TrustStoreType** This probe addresses the following Enhancement Requests: **RFE 71896:** Enhancement made to support the newest version of the Network Node Manager-i version (10.0). **RFE 65044:** Enhancement made to support HTTPS connection to the probe.
SC27-6551-02	June 28, 2019	"Summary" on page 1 updated for version 3 of this probe. Support for Network Node Manager-i version 2018.11 added.

Table 1. Document modification history (continued)		
Document version	**Publication date**	**Comments**
SC27-6551-03	June 26, 2020	"Summary" on page 1 updated for version 4 of this probe. Support for Network Node Manager-i version 2019.11 added. The following security enhancements which include: • Probe sanitizes event data for XML escaped characters. • Probe now validates HTTP host headers for `IncidentNotification` events. • Probe now disables TLS client-initiated renegotiations.

Conventions used in this guide

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

Operating system-dependent variables and paths

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

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

`$OMNIHOME/probes`

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

`%OMNIHOME%\probes`

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

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

Operating system-specific directory names

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

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

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

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

Table 2. Directory names for the arch variable	
Operating system	**Directory name represented by *arch***
AIX® systems	`aix5`
Red Hat Linux® and SUSE systems	`linux2x86`
Linux for System z	`linux2s390`
Solaris systems	`solaris2`
Windows systems	`win32`

OMNIHOME location

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

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

Chapter 1. Probe for Network Node Manager-i

The Network Node Manager-i (NNMi system) is a network management system that discovers network devices and provides a map of the network's topology. The system identifies the source of network problems and assists network managers in planning for network growth and designing network changes.

The subscribes to the NNMi system for event information and reports these as incidents to Netcool/OMNIbus. The probe can also synchronize its event information with the NNMi system at regular intervals. Both the subscription and resynchronization streams can be filtered. The probe can also be subscribe to the XML event descriptions that the NNMi API makes available.

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 NNMi.

Table 3. Summary

Probe target	Network Node Manager-i version 10.0 Network Node Manager-i version 10.10 Network Node Manager-i version 2018.11 Network Node Manager-i version 2019.11
Probe executable name	`nco_p_hp_nnm`
Patch number	4.0
Probe supported on	For details of supported operating systems, see the following Release Notice on the IBM Software Support website: https://www-304.ibm.com/support/docview.wss?uid=swg21687619
Properties file	`$OMNIHOME/probes/`*`arch`*`/hp_nnm.props`

Table 3. Summary (continued)	
Rules file	`$OMNIHOME/probes/arch/hp_nnm.rules`
Requirements	For details of any additional software that this probe requires, refer to the `description.txt` file that is supplied in its download package.
Connection method	HTTP/SOAP
Multicultural support	Available
Peer-to-peer failover functionality	Available
IP environment	IPv4 and IPv6
Federal Information Processing Standards (FIPS)	IBM Tivoli Netcool/OMNIbus uses the FIPS 140-2 approved cryptographic provider: IBM Crypto for C (ICC) certificate 384 for cryptography. This certificate is listed on the NIST website at http://csrc.nist.gov/groups/STM/cmvp/documents/140-1/1401val2004.htm. For details about configuring Netcool/OMNIbus for FIPS 140-2 mode, see the *IBM Tivoli Netcool/OMNIbus Installation and Deployment Guide*.

Getting started

This section shows how to start the probe with the minimum required configuration. The procedure assumes that you have a version of Netcool/OMNIbus installed and running.

Use the following procedure to start the probe with a minimal configuration:

Note : The commands shown in this example are for a Linux system. Adapt these commands as necessary for the operating system that your probe server runs.

1. Download the probe's installation package following the instructions in http://www-01.ibm.com/support/knowledgecenter/SSSHTQ/omnibus/probes/all_probes/wip/reference/install_download_intro.html.
2. Install the probe following the instructions in http://www-01.ibm.com/support/knowledgecenter/SSSHTQ/omnibus/probes/all_probes/wip/reference/install_install_intro.html.
3. Edit the probe's properties file and set values for the following: [insert nested list of properties and the values they need]
4. Ensure that `$NCHOME/etc/omni.dat` includes information on the ObjectServer.
5. If you made changes to `omni.dat`, regenerate the definition file by running `$NCHOME/bin/nco_igen`.
6. Check that the server where you have installed the probe can contact the ObjectServer using the `ping` utility. For example:

   ```
   ping -c 10 server-name
   ```

   Replace *server-name* with the name of the server where the ObjectServer is running.

   Alternatively, if you have installed the Netcool/OMNIbus desktop feature, you can use the `nco_ping` utility to check connectivity to the ObjectServer. For example:

   ```
   nco_ping object-server
   ```

   Replace *object-server* with the name of the ObjectServer.

7. Obtain a listing of the probe's command line options to check the probe is installed correctly:

   `$NCHOME/omnibus/probes/nco_p_hp_nnm -help`

   Check that the output from this command begins as follows:

```
Using IBM Java
Version 1.8.0
Usage: "nco_p_hp_nnm" [options]
where options can be:
 -autosaf                    Enable automatic Store and Forward on startup
 -beatinterval               Probe failover heartbeat interval
 -beatthreshold              Probe failover heartbeat threshold time
 -buffer                     Turn on alert buffering
 -bufferflushinterval        The interval (in seconds) before flushing the alert buffer
```

8. Start the probe as follows:

```
$NCHOME/omnibus/probes/nco_p_hp_nnm -propsfile $NCHOME/omnibus/probes/linux2x86/
hp_nnm.props -messagelevel debug
```

9. Check the probe's log file to ensure the probe started correctly and is ready to receive, process, and dispatch events.

   The probe's **MessageLog** property provides the name and location of the probe's log file.

10. Where possible, use a test tool to send events to the probe and so check they are processed correctly.

    For example, use the `curl` utility to send an event to the probe.

The probe is now successfully installed and operational. You can now configure the probe to suit your operating environment.

Installing probes

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

The installation process consists of the following steps:

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

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

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

2. Installing the probe using the installation package.

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

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

3. Configuring the probe.

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

Configuring the probe

After installing the probe you need to make various configuration settings to suit your environment. You also need to install NNMi utilities.

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

Table 4. Configuring the probe

Feature	Properties	See
Mandatory features:		
NNM system connection information Specifies how the probe subscribes to the NNMi system for events.	**NnmAddress** **NnmPort** **ProbeReceiverPort** **SubscriptionFilter**	"Connecting to the NNMi system" on page 7
Authentication Credentials for authenticating with the TMF endpoint.	**NnmPassword** **NnmUserId**	"Authentication" on page 8
ObjectServer host name	**Server**	"ObjectServer information" on page 6
Optional features:		
Event synchronization policy Specifies whether the probe synchronizes with the NNM system at startup and on regular occasions.	**InitialResync** **ResyncInterval** **ResyncBatchSize** **ResyncFilterName** **ResyncFilterOperation** **ResyncFilterValue**	"Event synchronization" on page 10
Reconnection policy Specifies whether the probe attempts to reconnect to the NNM system following a communications failure.	**RetryCount** **RetryInterval**	"Reconnection and backoff strategy" on page 12
Inactivity policy Specifies whether the probe disconnects from the NNM system following a period of inactivity.	**Inactivity**	"Inactivity" on page 12

Table 4. Configuring the probe (continued)		
Feature	**Properties**	**See**
Support for Unicode and non-Unicode characters Enables the probe to process alarms that contain characters encoded in UTF-8, such as Asian languages.	None	"Support for Unicode and non-Unicode characters" on page 12
Peer-to-peer failover pair Allows you to set up two probes to act as a failover pair to improve availability. If the master probe should stop working, the slave probes takes over until the master is available once more.	**MessageFile** **Mode** **PeerHost** **PeerPort** **PidFile** **PropsFile** **RulesFile**	"Peer-to-peer failover functionality" on page 13

Installing the NNMi SDK JAR libraries

The NNMi SDK JAR libraries must be installed in order for the probe to connect with the NNMi.

To install these JAR file, use the following steps:

1. Create the following directory on the probe system:

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

2. Copy the `nms-sdk.jar` file from the *$NNM_HOME*`/OV/NNM/lib` directory located on the NNMi system to the *$OMNIHOME*`/probes/java/nco_p_hp_nnm` directory on the probe system.

   Note : The *$NNM_HOME* variable specifies the location where NNMi is installed.

3. Ensure that the default location setting for the utility JAR file exists on the respective operating system platforms:

   - **On Windows:**

     File:

     ```
     %OMNIHOME%\probes\win32\nco_p_hp_nnm.bat
     ```

     Where the default configuration is:

     ```
     set PROBE_CLASSPATH=%CLASS_DIR%\nco_p_hp_nnm.jar;%OMNIHOME%\probes\
     java\nco_p_hp_nnm\nms-sdk.jar
     ```

   - **On Unix:**

     File:

     ```
     $OMNIHOME/probes/java/nco_p_hp_nnm.env
     ```

     Where the default configuration is:

     ```
     CLASSPATH_SETTING=${OMNIHOME}/probes/java/nco_p_hp_nnm/nms-sdk.jar
     ```

Running the probe

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

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

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

Data acquisition

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

The Probe for NNMi acquires data from NNMi.

Data acquisition is described in the following topics:

ObjectServer information

The probe sends processed events to an ObjectServer resource. This can be a single server or a failover pair of ObjectServers.

The probe sends processed events to an ObjectServer resource. This can be a single server or a failover pair of ObjectServers.

Single ObjectServer

Configure the probe to communicate with an ObjectServer by setting the **Server** property to the name of the ObjectServer.

Failover pair

Optionally, you can define a failover pair of ObjectServers. One of them acts as the primary ObjectServer that the probe communicates with initially. Should that ObjectServer become unreachable, the probe communicates with the other, backup ObjectServer. To define a failover pair set the following probe properties:

Table 5. Properties that define a failover pair of ObjectServers

Property	Value
Server	Set this property to the name of the primary ObjectServer.
ServerBackup	Set this property to the name of the backup ObjectServer.

Table 5. Properties that define a failover pair of ObjectServers (continued)	
Property	**Value**
NetworkTimeout	Set this property to the timeout period, in seconds, for the primary ObjectServer. If the probe does not receive a response from the primary ObjectServer within that time, it connects to the backup ObjectServer.
PollServer	Set this property to the time period, in seconds, when the probe tries to reconnect to the primary ObjectServer.

Note : Ensure that the value of **NetworkTimeout** is less than the value of **PollServer**.

Connecting to the NNMi system

The probe connects to the system and subscribes to the notification event service to receive new events as they are generated.

When it starts, the probe connects to the NNMi system using HTTP/HTTPS with the following properties:

- **NNMAddress**: This property specifies the hostname or IP address of the NNMi system in IPv4 or IPv6 format, as appropriate for your network.
- **NNMPort**: The probe uses this property to specify the port on the NNMi system where it receives subscription requests.

The administrator of the NNMi system can provide the values to use for these properties.

Once the connection is active, the probe sends a SOAP message that creates a subscription for receiving events from the NNMi system. This message includes the identity of the port on the host that runs the probe where it receives events. The **ProbeReceiverPort** identifies this port. Set this to a suitable value for your environment.

In this configuration, the probe starts an endpoint to which the NNMi system sends the events.

The subscription filter

The SOAP message that creates the event subscription can include a subscription filter. The filter defines the events that the probe wishes to receive from the NNMi system. For example, the probe can specify that it wants to receive events with a severity of critical only.

Use the **SubscriptionFilter** property to define subscription filter. The property's value is an XPath expression that identifies the events that the probe wants to receive. The following are some examples of the subscription filter:

Table 6. Examples of the subscription filter	
Value of the SubscriptionFilter property	**Description**
`"arg0[severity='CRITICAL']"`	The probe receives events that have a severity of critical only.
`"arg0[origin='MANAGEMENTSOFTWARE']"`	The probe receives events that originate from the `MANAGEMENTSOFTWARE` component only.

By default, the subscription filter is empty, and so the probe receives all events that the NNMi system generates.

Firewall considerations

The **ProbeReceiverPort** property defines the port where the probe receives messages from the NNMi system. Ensure that port is open to receive messages on the system that runs the probe.

The **NnmPort** property defines the port on the NNMi system that the probe communicates with. Ensure that port is open on the NNMi system.

Creating a secure channel

The probe uses secure transport layer (TLS) to enable a secured HTTPS transmission. You can choose to use HTTP (non-secured) or HTTPS by configuring the **EnableSSL** property and setting the keystore/truststore.

Configuring secure connections

If the server is using a TLS connection to encrypt data exchanged over JMS and HTTP, you must configure the truststore for the HTTPS connection on the Tivoli Netcool/OMNIbus probe server.

Authentication

When subscribing to events from the NNMi system, the probe provides authentication information in the form of a username and password.

Use the **NNMUserID** and **NNMPassword** properties to specify the credentials for the account on the NNMi system that the probe uses for event notification. The values of both properties can be plain text or AES encrypted text. However, for improved security encrypt both the user name and password.

To encrypt the username or password, use the **nco_keygen** utility to create a key file and then use the **nco_aes_crypt** utility to encrypt the text using the key file.

Detailed instructions on how to encrypt a property value, such as **NNMUserID** and **NNMPassword** are in the *IBM Tivoli Netcool/OMNIbus Installation and Deployment Guide*. The following example summarizes the procedure for encrypting the password:

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

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

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

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

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

   ```
   ConfigCryptoAlg: "AES"
   ```

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

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

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

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

Event retrieval

When the probe is running, it receives events from the NNMi system as they are generated. The probe receives details of new incident events and updates to existing events that match the criteria in the **SubscriptionFilter** property.

Each message from the NNMi system contains a `$Resent`, `$Identifier` and `$UpdateTime` element. These elements indicate whether the message is new, an update to a previous message, or if the message has previously been processed by the probe.

For existing events, the probe receives notification only if either the `$LifecycleState` or `$RcaState` element of the incident is updated.

The probe maintains an internal cache of events it has received. The probe uses this cache to avoid sending false duplicate events to the Netcool/OMNIbus event list.

The NNMi system can detect if a message has not been sent, and makes three attempts to resend the message before it is discarded.

Subscription checking and renewal

The probe periodically checks the status of and renews its subscription with the NNMi system.

Checking the subscription

The probe regularly checks whether the subscription with the NNMi server is active. The value of the **`HeartbeatInterval`** property determines the frequency (in seconds) of this check.

The default value of **`HeartbeatInterval`** is 60. Change this, if necessary, to suit your local environment. The value of the property can be in the range 1 to 299. Note that the value of this property also affects how often the probe renews the subscription (see "Renewing the subscription" on page 9). Take that into account when determining the value of the property that is suitable for your site. For more details, refer to "Configuring a secure connection with the NNMi system" on page 14.

Note : The probe will not successfully start up if the **`HeartbeatInterval`** property is not configured correctly.

Renewing the subscription

In addition to checking the subscription, the probe also renews it. The value of the **`HeartbeatInterval`** property has a role in determining the expiry time of the subscription:

Table 7. Determining the subscription expiry time

Value of HeartbeatInterval	Subscription expiry time
<=240	The probe sets the subscription expiry time to 5 minutes (300 seconds).
>240	The probe sets the subscription expiry time to the value of **`HeartbeatInterval`** plus one minute (60 seconds). For example, if the value of **`HeartbeatInterval`** is 270, the probe sets the subscription expiry time to: 270 + 60 = 330 seconds = 5 minutes 30 seconds

The initial subscription expiry time

The probe creates the initial subscription when it connects to the NNMi system. The expiry time of this subscription is always 5 minutes, which is the default expiry time of the NNMi system.

Failure of the subscription check

The subscription check can fail for a number of reasons, which includes:

- The NNMi system is not running
- The NNMi system has become unreachable as the network connection to the NNMi system has failed or is unavailable.

If the subscription check fails, the probe does one of the following:

- Immediately shuts down
- Attempts to reconnect to the NNMi system

The value of the **RetryCount** property determines which action the probe takes. When the property value is 0, the probe immediately shuts down. When the property has a positive value, the probe attempts to reconnect to the NNMi system as shown in "Reconnection and backoff strategy" on page 12.

Event synchronization

Synchronization enables the probe to collect any events that the NNMi system generated while the probe was disconnected. In addition, the probe can resynchronize with the NNMi system at regular intervals. You can use any combination of these synchronization features.

Initial synchronization

Each time the probe starts it can synchronize with the NNMi system to acquire all the events that the NNMi system holds. Use the **InitialResync** property to control whether the probe carries out an initial synchronization, but instead performs a resynchronization at the interval defined by the **ResyncInterval** property.

When the value of the property is `true`, the probe carries out an initial synchronization. When the value of the property is `false`, the probe does not carry out an initial synchronization.

By default, the probe does not synchronize with the NNMi system at startup.

Synchronization at intervals

In addition to synchronizing with the NNMi system at startup, the probe can also resynchronize with the NNMi system at regular intervals. Use the **ResyncInterval** property to define whether the probe synchronizes with the NNMi system at intervals and how frequently.

When the value of the property is 0 or less, the probe does not resynchronize with the NNMi system at regular intervals. When the value of the property is a positive number, the probe resynchronizes with the NNMi system regularly. The value of the property specifies the number of seconds between each resynchronization operation.

By default, the probe does not regularly resynchronize with the NNMi system.

Synchronization in batches

You can use the **ResyncBatchSize** property to specify the maximum number of alarms contained in each batch that the probe receives during the resynchronization operation. There may be multiple batches within one resynchronization operation.

The default value of this property is 100.

Defining a resynchronization filter

By default, a synchronization operation retrieves all the events that the NNMi server currently holds. You can use a resynchronization filter to limit the number of events that the event retrieves. Three properties contain the resynchronization filter:

ResyncFilterName
ResyncFilterOperation
ResyncFilterValue

ResyncFilterName

ResyncFilterName identifies a field in the event that you want to filter on. You can specify the following:

- assignedTo

- category
- cias
- created
- duplicateCount
- family
- firstOccurrenceTime
- formattedMessage
- id
- lastOccurrenceTime
- lifecycleState
- modified
- nature
- name
- notes
- origin
- originOccurrenceTime
- priority
- rcaActive
- severity
- sourceName
- sourceNodeName
- sourceNodeUuid
- sourceType
- sourceUuid
- uuid

ResyncFilterOperation

ResyncFilterOperation defines a condition to use in the filter. You can use the following operators:

- =
- !=
- <
- <=
- >
- >=
- LIKE
- NOT IN

ResyncFilterValue

ResyncFilterValue defines the value of the field identified in **ResyncFilterName** to be compared against the corresponding field in each event using the operation in **ResyncFilterOperation**.

Example

To retrieve only those events that have a severity of Error or greater, use the following property values:

```
ResyncFilterName: "severity"
ResyncFilterOperation: ">="
ResyncFiltervalue: "ERROR"
```

Optimized Synchronization

During a resynchronization operation, the probe tries to optimize the process by only retrieving events that it has not retrieved previously. The probe does this using the time stamp associated with each event.

Reconnection and backoff strategy

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

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

Use the **RetryInterval** property to specify the number of seconds between each attempt to reconnect to the NNMi system.

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

Inactivity

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

The **Inactivity** property specifies the maximum time (in seconds) that the probes waits, without receiving any messages from the NNMi system. If that amount of time passes by, the probe shuts down.

To ensure that the probe never shuts down because of inactivity, set the property to 0.

Support for Unicode and non-Unicode characters

The probe can process multibyte characters such as Unicode UTF-8, GB, Big5, or Shift-JIS.

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

1. Ensure that the NNMi system is configured to send data in the required format (for example, UTF-8).
2. Set the required locale on the system that runs the probe:

Table 8. Setting the locale for multibyte characters

Operating system	Procedure to set the locale

Linux and Unix	Set the locale by changing the values of the **LANG** and **LC_ALL** environment variables. For example, to set the locale to simplified Chinese in UTF-8, use the following commands:
	`export LANG=zh_CN.UTF-8`
	`export LC_ALL=zh_CN.UTF-8`

Table 8. Setting the locale for multibyte characters (continued)	
Operating system	**Procedure to set the locale**
Windows	a. Open the Control Panel and double click on **Regional and Language**. b. On the **Formats** tab, select the language from the list in **Format**. c. On the **Administrative** tab, click **Change system locale**. d. Select the language from the list in **Current System Locale**. e. Click **OK**. f. Click **OK**.

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

   When running the probe on a Windows system using a UTF-8 character set, always specify the `-utf8enabled` command line option. For all other character sets, **do not** use the `-utf8enabled` command line option.

Peer-to-peer failover functionality

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

While the slave probe receives heartbeats from the master probe, it does not forward events to the ObjectServer. If the master probe shuts down, the slave probe stops receiving heartbeats from the master and any events it receives thereafter are forwarded to the ObjectServer on behalf of the master probe. When the master probe is running again, the slave probe continues to receive events, but no longer sends them to the ObjectServer.

Example property file settings for peer-to-peer failover

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

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

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

```
Server      :    "NCOMS"
RulesFile   :    "master_rules_file"
MessageLog  :    "master_log_file"
PeerHost    :    "slave_hostname"
PeerPort    :    6789 # [communication port between master and slave probe]
Mode        :    "master"
PidFile     : "master_pid_file"
ProbeReceiverPort : 8080 # [unique communication port between the probe and the nnmi server]
```

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

```
Server      :    "NCOMS"
RulesFile   :    "slave_rules_file"
MessageLog  :    "slave_log_file"
PeerHost    :    "master_hostname"
PeerPort    :    6789 # [communication port between master and slave probe]
Mode        :    "slave"
```

```
PidFile     : "slave_pid_file"
ProbeReceiverPort : 8081 # [unique communication port between the probe and the nnmi server]
```

Configuring a secure connection with the NNMi system

This section describes how to configure the probe to securely connect to the NNMi system using HTTPS.

Use the following steps to prepare the probe for deployment:

1. Identify the install `$NnmInstallDir` and data `$NnmDataDir` directories on the NNMi system. The location of these directories can be identified from the NNMi console under the following menu and tabs:

   Help --> System Information --> Server (Tab)

   The keystore and truststore files are located in the `$NnmDataDir` directory.
2. Create a truststore file for the probe using the following keytool command:

   ```
   keytool -genkey -keyalg RSA -alias OMNIKEY -keystore /root/nnmi_truststore.jks -storetype
   jks -storepass password -validity 360 -keysize 2048
   ```
3. Extract the NNMi certificate from the NNMi keystore using the following command:

   ```
   keytool -export -alias nnmi -keystore /root/nnm-key.p12 -storepass nnmkeypass -file /root/
   mynnm.cert
   ```
4. Import the certificate into the probe truststore and copy the truststore file to the probe system using the following command:

   ```
   keytool -import -trustcacerts -alias nnmi -file /root/mynnm.cert -keystore /root/
   nnmi_truststore.jks
   ```
5. Configure the following probe properties on the probe system:

   ```
   TrustStore           : '<path to truststore file>'
   TrustStorePassword   : '<truststore password>'
   TrustStoreType       : 'JKS'

   NNMAddress           : <NNMi hostname which matches the Common Name (CN) from the NNMi
   certificate>
   NNMPort              : 443
   NNMUserID            : '<username>'
   NNMPassword          : '<password>'
   ```

Properties and command line options

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

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

Note : The **ProcessSNMPAsV1** property that was used in Probe for NNMi Version 1.0 has been removed. SNMP messages from different versions are uniformly represented using a common data object for the uniformed extraction process.

Table 9. Properties and command line options		
Property name	**Command line option**	**Description**
EnableSSL *string*	`-ssl` (This is equivalent to **EnableSSL** with a value of `true`.) `-nossl` (This is equivalent to **EnableSSL** with a value of `false`.)	Use this property to specify whether the probe enables an SSL connection between the probe and NNMi server using the HTTPS protocol. The default is `false`
EnableSSLDebug *string*	`-ssldebug` (This is equivalent to **EnableSSLDebug** with a value of `true`.) `-nossldebug` (This is a equivalent to **EnableSSLDebug** with a value of `false`.)	Use this property to specify that the probe enables verbose logging on SSL. The default is `false`. **Note :** Only set this property to true for debugging purposes.
KeyStore *string*	`-keystore` *string*	Use this property to specify the file name (with full path) of a keystore that contains the OMNIbus probe certificate and private key for SSL if client authentication is required by NNMi server. This keystore is expected to contain only one key entry. The default is `""`.
KeyStorePassword *string*	`-keystorepassword` *string*	Use this property to specify the password required to access the keystore specified by the **KeyStore** property. The default is `""`.
KeyStoreType *string*	`-keystoretype` *string*	Use this property to specify the type of keystore specified by the **KeyStore** property. Available options are JKS, PKCS12, and any other keystore type supported by JRE. The default is JKS.

Table 9. Properties and command line options (continued)

Property name	Command line option	Description
NNMAddress *string*	-nnmip *string*	The hostname or IP address of the NNMi server instance to be monitored. If the **EnableSSL** property is set to true, the value of this property should match with the value of CN field or subject alternative names field in NNMi certificate for the proper host name/IP address verification during SSL handshaking. The default is 127.0.0.1.
NNMPassword *string*	-nnmpassword *string*	Use this property to specify the encrypted password to login to the NNMi server web services. The default is "".
NNMPort *integer*	-nnmport *integer*	Use this property to specify the port on the NNMi system where the system's web services are exposed. The default is 80.
NNMUserID *string*	-nnmusername *string*	Use this property to specify the encrypted user ID to login into NNMi server web services. The default is "".
ProbeReceiverPort *integer*	-probereceiverport *integer*	Use this property to specify the port that receives events from the NNMi system. The default is 8080.
ResyncBatchSize *integer*	-resyncbatchsize *integer*	Use this property to specify the number of events contained in each batch request. The default is 100.
ResyncFilterName *string*	-resyncfiltername *string*	Use this property to specify the name of the attribute to be acted on by the resynchronization filter. The default is severity.

Table 9. Properties and command line options (continued)		
Property name	**Command line option**	**Description**
ResyncFilterOperation *string*	`-resyncfilteroperation` *string*	Use this property to specify the condition to filter events on during a resynchronization operation. The operation to be applied in resynchronization can take the following values: =, !=, >, >=, <, <=. The default is `"="`.
ResyncFilterValue *string*	`-resyncfiltervalue` *string*	Use this property to specify the value to be checked for by the resynchronization filter. The default is `"CRITICAL"`.
SecurityProtocol *string*	`-securityprotocol`*string*	Use this property to specify which secure socket protocol the probe uses when SSL encryption is enabled. The value of this property cannot be emptied. The actual available possible values for this property depends on the implementation of the JRE that the probe is run with, as well as the security protocols supported by the current version of the connected NNMi system. The default is TLSv1.
SubscriptionFilter *string*	`-subscriptionfilter` *string*	Use this property to specify the XPath string to describe the filter for subscription events. The default is `""`.
TrustStore *string*	`-truststore` *string*	Use this property to specify the full path to the truststore file containing the trusted Certificate Authority certificate for the NNMi system. This property must be set if the **EnableSSL** property is set to `true`. The default is `""`.
TrustStorePassword *string*	`-truststorepassword` *string*	Use this property to specify the password required to access the truststore file specified by the **TrustStore** property. The default is `""`.

Table 9. Properties and command line options (continued)		
Property name	**Command line option**	**Description**
TrustStoreType *string*	`-truststoretype` *string*	Use this property to specify the type of truststore specified by **TrustStore** property. Available options are JKS, PKCS12, and any other keystore type supported by JRE. The default value is JKS.

Properties and command line options provided by the Java Probe Integration Library (probe-sdk-java) version 8.0

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 8.0.

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

Table 10. 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 `60`.

Table 10. 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).
InitialResync *string*	`-initialresync` *string*	Use this property to specify whether the probe requests all active alarms from the host server on startup. This property takes the following values: `false`: The probe does not request resynchronization on startup. `true`: The probe requests resynchronization on startup. For most probes, the default value for this property is `false`. If you are running the JDBC Probe, the default value for the **InitialResync** property is `true`. This is because the JDBC Probe only acquires data using the resynchronization process.
MaxEventQueueSize *integer*	`-maxeventqueue size`*integer*	Use this property to specify the maximum number of events that can be queued between the non native process and the ObjectServer. The default is `0`. **Note :** You can increase this number to increase the event throughput when a large number of events is generated.
ResyncInterval *integer*	`-resyncinterval` *integer*	Use this property to specify the interval (in seconds) at which the probe makes successive resynchronization requests. The default value for this property is `0` (which instructs the probe to not make successive resynchronization requests).
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).

Table 10. Properties and command line options (continued)		
Property name	**Command line option**	**Description**
RetryInterval *integer*	`-retryinterval` *integer*	Use this property to specify the length of time (in seconds) that the probe waits between successive connection attempts to the target system. The default is `0` (which instructs the probe to use an exponentially increasing period between successive connection attempts, for example, the probe will wait for 1 second, then 2 seconds, then 4 seconds, and so forth).

Elements

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

The following table describes the elements that the Probe for NNMi generates. Not all the elements described are generated for each event; the elements that the probe generates depends upon the event type. For more detailed information about each token, see the NNMi documentation.

Table 11. Elements	
Element name	**Element description**
`$Id`	This element contains the ID of the NNMi user.
`$AssignedTo`	This element identifies the user to whom the alarm was assigned
`$Category`	This element shows the category priority of the alarm.
`$cia_address`	This element contains the custom incident attributes (CIA) address of the source from which the alarm was sent.
`$cia_agentAddress`	This element contains the CIA SNMP agent address.
`$cia_originaladdress`	This element identifies the original trap's address.
`$cia_snmpoid`	This element identifies the CIA SNMP object identifier (OID).
`$DuplicateCount`	This element contains the number of times the alarm has been duplicated.
`$Family`	This element identifies the event source.
`$FormattedMessage`	This element contains the name of the alarm.
`$LifecycleState`	This element displays the lifecycle status of the token generated.

Table 11. Elements (continued)	
Element name	**Element description**
$Name	This element contains the name of the event.
$Notes	This element identifies user supplied remarks/notes.
$Priority	This element indicates the priority of the alarm.
$SourceName	This element contains the name of the alarm source.
$SourceNodeName	This element displays the node name of the source from which the alarm was sent.
$SourceNodeUuid	This element displays the node UUID of the source from which the alarm was sent.
$Severity	This element indicates the severity of the alarm. Possible values include: • CRITICAL • MAJOR • MINOR • WARNING • NORMAL
$Uuid	This element shows the unique identifier for an NNMi event.
$Created	This element identifies the time the event was created.
$FirstOccurrenceTime	This element identifies the first occurrence of the event.
$LastOccurrenceTime	This element identifies the last time the event occurred.
$Modified	This element identifies the last time the alarm's details were modified.
$Nature	This element indicates the nature of the alarm. Possible values include: • ROOTCAUSE • SECONDARYROOTCAUSE • SYMPTOM • SERVICEIMPACT • STREAMCORRELATION • NONE

Table 11. Elements (continued)	
Element name	**Element description**
$Origin	This element indicates the origin of the alarm. Possible values include: • MANAGEMENTSOFTWARE • MANUALLYCREATED • REMOTELYGENERATED • SNMPTRAP • SYSLOG • OTHER
$OriginOccurrenceTime	This element identifies the time of the occurrence that generated the event.
$SourceType	This element identifies the data type of the source object.

Error messages

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

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

Table 12. Error messages

Error	Description	Action

`Encountered a JAXException attempting to send subscription request`
`Encountered a SOAPException attempting to send subscription request`
`Encountered a WebServiceException attempting to send subscription request`
`Error creating a DocumentBuilder instance: message`
`Error creating a JAXBContext instance: message`
`Error formatting Expires to String: message`
`Error parsing Expires to Date: message`
`ReferenceParameter does not exist in SubscribeResponse`
	These errors indicate problems with probe startup.	Ensure that the**NNMPort** property points to the port where the NNMi system has been configured to host the WS-API.
		Check any firewalls between the probe and the NNMi machine.
		Confirm that the NNM WebServices API is functioning correctly by checking that your NNMi installation has an Integration Enablement License and that the services are available. This can be confirmed by looking at the jmx-console view of the NNM Jboss server.
		Check that the environment has been configured to use Java.

`HeartbeatInterval out of range! Please reconfigure property.`	The probe performed a check on the value specified in the **HeartbeatInterval** property and determined it was out of range.	Check the value of the **HeartbeatInterval** property an ensure the value is between 1 and 299.

`NNM credentials not recognized`
`End Point Started at : Location of Probe's Receiver Endpoint`
`End Point Stopped at : Location of Probe's Receiver Endpoint`
	These errors indicate problems with the probe configuration.	Ensure that the **NNMAddress**, **NNMPassword**, and **NNMPort** properties are set correctly.
		Check that the Probe Receiver Port is clear.

Table 12. Error messages (continued)

Error	Description	Action
`Unable to contact the NNM Incident service for resynchronization because: message`	The probe was unable to contact the NNMi Incident service for resynchronization. The *message* provides additional information about the error.	Check the values of the **NNMAddress** and **NNMPort** properties to ensure a valid TCP/IP address and port number for the NNMi system. Also check that there is no firewall blocking access to the NNMi system.
`Unable to process incident array received in resynchronization because: message`	The probe was unable to process the NNMi incident array received in the resynchronization operation. The *message* provides additional information about the error.	Check the values of the **NNMUserID** and **NNMPassword** properties to ensure these are valid encrypted identifiers of an account on the NNMi system. Also check that the value specified for the **ResyncFilterValue** property is valid for the event values specified in the **ResyncFilterName** property.
`Unable to process requested custom filter`	The probe was unable to process the custom filter. As a result, the probe continues processing with the default resynchronization filters.	A resynchronization filter was defined using the **ResyncFilterName**, **ResyncFilterOperation**, and **ResyncFilterValue** properties. Check each of these properties as follows: • That the name you specified for the **ResyncFilterName** is one of the valid names. Note that these names are case sensitive. • That the operation you specified for the **ResyncFilterOperation** property is one of the valid operators and conditions. The character operators are case sensitive. See "Defining a resynchronization filter" on page 10 for the valid values associated with these properties.

Appendix A. Notices and Trademarks

This appendix contains the following sections:

- Notices
- Trademarks

Notices

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

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

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

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

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

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

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

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

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

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

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

IBM Corporation
Software Interoperability Coordinator, Department 49XA

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

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

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

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

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

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

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

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

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

COPYRIGHT LICENSE:

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

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

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

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

Trademarks

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

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

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

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

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

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

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

SC27-6551-03