IBM® Tivoli® Netcool/OMNIbus Probe for HPE Operation Manager i 1.0

Reference Guide
December 14, 2017

SC27-8772-00

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 (SC27-8772-00) applies to version 1.0 of IBM Tivoli Netcool/OMNIbus Probe for HPE Operations Manager i and to all subsequent releases and modifications until otherwise indicated in new editions.

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 Probe for HPE Operations Manager documentation is provided in softcopy format only. To obtain the most recent version, visit the IBM® Tivoli® Knowledge Center:

https://www.ibm.com/support/knowledgecenter/SSSHTQ/omnibus/probes/common/Probes.html

Table 1. Document modification history

Document version	**Publication date**	**Comments**
SC27-8772-00	December 14, 2017	First IBM publication.

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 HPE Operations Manager i

IBM Tivoli Netcool/OMNIbus Probe for HPE Operations Manager i (HPE OMi) can acquire XML-formatted events from HPE OMi using the REST API. It converts these events into Netcool/OMNIbus events and sends them to the ObjectServer.

Note : The probe is only supported on OMNIbus V8.1 and above.

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 HPE Operations Manager i.

Table 3. Summary

Probe target	HPE Operations Manager i version 10.60 and above.
Probe executable name	`nco_p_hpe_omi`
Probe installation package	`omnibus-`*`arch`*`-probe-nco-p-hpe_omi-`*`version`*
Package version	1.0
Probe supported on	For details of supported operating systems, see the following Release Notice on the IBM Software Support website: http://www-01.ibm.com/support/docview.wss?uid=swg22011350
Properties file	`$OMNIHOME/probes/`*`arch`*`/hpe_omi.props`
Rules file	`$OMNIHOME/probes/`*`arch`*`/hpe_omi.rules`
Transport Module properties file	`$OMNIHOME/java/conf/hpeomiTransport.properties` **Note :** The probe also requires Transport Module version 17 (`common-transportmodule-17_0`).

Table 3. Summary (continued)

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	REST API
Multicultural support	Not Available
Peer-to-peer failover functionality	Available
IP environment	IPv4 and IPv6
Federal Information Processing Standards (FIPS)	IBM Tivoli Netcool/OMNIbus uses the FIPS 140-2 approved cryptographic provider: IBM Crypto for C (ICC) certificate 384 for cryptography. This certificate is listed on the NIST website at http://csrc.nist.gov/groups/STM/cmvp/documents/140-1/1401val2004.htm. For details about configuring Netcool/OMNIbus for FIPS 140-2 mode, see the *IBM Tivoli Netcool/OMNIbus Installation and Deployment Guide*.

Installing probes

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

The installation process consists of the following steps:

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

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

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

2. Installing the probe using the installation package.

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

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

3. Configuring the probe.

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

4. Copy `opr-external-api.jar` and `xercesImpl-2.9.1.jar` to the `$OMNIHOME/probes/<arch>` directory.

   These jar files can be obtained from the OMi environment.

Configuring JRE

The probe requires JRE 1.8 or above.

Install the required JRE if not already done. Netcool OMNIbus 8.1 is bundled with JRE 1.7, so you will download JRE 1.8 if your system does not have it.

After the JRE installation, configure `JAVA_HOME` to include the path to that JRE.

SSL-based connectivity

The Probe for HPE Operations Manager i supports Secure Sockets Layer (SSL) connections between the probe and HPE Operations Manager i. SSL connections provide additional security when the probe retrieves alarms from the target systems.

To enable SSL connections, obtain the required SSL certificates and the Trusted Authority certificate from the HPE Operations Manager i server administrator. Add the certificates to a local Java™ keystore so that they can be referenced by the **`KeyStore`** property.

Prerequisites

The following tools are available to create the keystore:

- The OpenSSL toolkit.

  This is available from http://www.openssl.org/.
- The IBM KeyMan utility.

  This is available from http://www.alphaworks.ibm.com/tech/keyman/download.
- The Keytool toolkit.

  This is available in the JRE package.

Converting the key and certificate into PKCS12 format

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

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

Where

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

`cert_file` is the certificate retrieved from the server.

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

Creating the SSL keystore

You can create a Java keystore using either the `KeyMan` utility or the `Keytool` utility.

To create a Java keystore using the `KeyMan` utility, follow these steps:

1. Start the `KeyMan` utility.
2. Click **Create New** and select the **Keystore token** option.
3. Click **File** > **Import** and choose the certificate that you retrieved from the server.

   This imports the certificate into the keystore.
4. Click **File** > **Save** and enter a password and name for the keystore; for example, *`trusted_keystore`*`.jks`.

To create a Java keystore using the `Keytool` utility, follow these steps:

1. Generate a keystore and self-signed certificate using the following command:

   ```
   keytool -genkey -keyalg RSA -alias alias_name -keystore keystore_file -
   storepass keystore_password -validity 360 -keysize 2048
   ```
2. Import the SSL certificate into the newly created Java keystore file using the following command:

   ```
   keytool -import -trustcacerts -alias alias_name -file cert_file -keystore
   keystore_file
   ```
3. Verify that the certificates are in a Java keystore using the following command:

   ```
   keytool -list -v -keystore keystore_file
   ```

Enabling SSL connections

To enable SSL-based connections between the probe and the Element Management System (EMS) server, make the following changes to the `hpe_omi.props` file:

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

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

   - **KeyStore**
   - **KeyStorePassword**
2. Use the **KeyStore** property to specify the location of the keystore file.
3. Use the **KeyStorePassword** property to specify a password for the keystore.

   Note : You can encrypt the keystore file password using the `nco_aes_crypt` utility (for FIPS 104-2 mode security).
4. Set the **Port** property to the port that the probe uses for SSL connections.

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

The probe acquires `OprEvent`, `OprEventList`, `OprEventChange`, and `OprEventChangeList` message objects in XML format from HPE Operations Manager i using the Transport Module. The probe converts the XML data into Netcool/OMNIbus events and sends them to the ObjectServer.

Data acquisition is described in the following topics:

- "Communicating using REST APIs and Webhook" on page 4
- "Configuring the transport module" on page 5
- "Event caching" on page 6
- "Data stream capture" on page 11

Communicating using REST APIs and Webhook

The Probe for HPE Operations Manager i (OMi) uses the REST API to send HTTP requests or secure HTTPS requests.

The probe uses the REST API to perform the following functions:

- Request alert resynchronization. This request returns the `OprEventList` object.
- Request full message. This request returns the `OprEvent` object.

The probe uses a Webhook to listen to the OMi event forwarder for alert notifications and updates. From this channel data can come in the following forms:

- `OprEvent`
- `OprEventChange`
- `OprEventList`
- `OprEventChangeList`

The `List`-typed objects are expected only when the OMi event forwarder is enabled with bulk transferring. The bulk configuration is controlled in the OMi system `Connected Server` instance meant for probe integration.

`OprEventList` holds a list of `OprEvent` objects, and `OprEventChangeList` holds a list of `OprEventChange` objects.

The key difference between `OprEvent` objects and `OprEventChange` objects is that the former contains full information of about the alert, and the latter contains only changed data (such as the time changed and the relevant updates) and referential information about the original alert.

The gap between these objects demands the use of an Event Cache to store the fields from the full originating alert. These fields are used by the probe rules processing to complement those available in the `OprEventChange`object. For details see "Event caching" on page 6.

The alert operations with the target system are configured by the following properties in the `hpeomiTransport.properties` file:

Table 4. Transport properties

Operation	**Transport module properties**
Resynchronization	**`resyncRequestHeaders`** **`resyncRequestURI`** **`resyncRequestMethod`**
Notification	**`webhookURI`**

For details about these properties, see "Configuring the transport module" on page 5.

Configuring the transport module

The transport properties file defines how the probe receives events from the transport module.

HPE OMi transport properties file

The probe is packaged with the `hpeomiTransport.properties` file which is a pre-configured transport properties file. To specify a different transport properties file, use the **`TransportFile`** property in the `hpe_omi.props` file.

The following table describes the properties used to configure the `hpeomiTransport.properties` file.

Table 5. HPE OMi transport properties	
Property name	**Description**
`resyncRequestHeaders`	Use this property to configure the headers in the HTTP request. The default is `Authorization=Basic ++Username++:++Password++,content-type=application/xml` `++Username++` and `++Password++` are resolved during runtime with the values configured in the **`Username`** and **`Password`** probe properties. The values must be consistent with the AUTH configuration in the OMi `Connected Server` instance.
`resyncRequestURI`	Use this property to specify the URI that the probe uses to request OMi alerts. The default is `/opr-web/rest/event_list`.
`resyncRequestMethod`	Use this property to specify the request method to retrieve alerts from the target system. The default is GET
`webhookURI`	Use this property to specify the URI that the probe uses to listen for notification forwarded from the target system. The value must be consistent to the URI configured in the OMi Connected Server instance. The default is `/probe/webhook`

Event caching

During probe rules processing, the OMi tokens involved in event correlation and deduplication must be present in every iteration.

An `OprEvent` object has all the tokens required in the rules computation. However, an `OprEventChange` object never contains all the tokens, and so the probe must request from OMi the full message (`OprEvent` object) of the original alert for each `OprEventChange` object. The HTTP operation would amount to substantial overhead if too frequently occur. To minimize the processing overhead, the probe stores the selected fields from the `OprEvent` object in the Event Cache for future reference to complement each `OprEventChange` object.

For the probe to manage the Event Cache using the following example configuration, specify the file holding these values in the **`EventCacheConfig`** probe property.

Table 6. Example configuration	
Field with default value	**Description**
`MAX_NODES=10000`	The maximum number of nodes in the cache.

Table 6. Example configuration (continued)

Field with default value	**Description**
`MSG_REQUEST_RERTY=3`	The number of retry to request `OprEvent` for the in-waiting event node.
`MSG_WAIT_TIMEOUT=30` (seconds)	The interval of an in-waiting event node.
`NODE_DURATION=20` (minutes)	The interval of an `OprEvent` node to stay in the cache.
`RENEW_DURATION=true`	To extend node duration if the node ever receives update.
`STORED_FIELDS=id, state, severity, priority, application object, key, originating_server, sending_server, time_changed, node_hints, title`	The selected fields from `OprEvent` to be cached.

For an `OprEventChange` object without a cached `OprEvent` node for reference, it enters the Event Cache as an in-waiting event node anticipating the full message returned from the probe's request to the OMi.

There are two possible outcomes to a full message request:

- If the full message is not obtained within the interval of `MSG_WAIT_TIMEOUT`, then the node is removed and the `OprEventChange` is sent to the probe rules as a discarded event as with a `$Discarded` token.
- If the full message is obtained within the interval of `MSG_WAIT_TIMEOUT`, then the in-waiting node becomes an `OprEvent` node with a full `NODE_DURATION`.

To maintain data consistency of the `OprEvent` nodes in the cache, whenever encountering its corresponding `OprEventChange` or `OprEvent`, the cached fields common to the object's fields will get the update. In return, the `OprEventChange` is complemented with the cached data, and assigned an additional `$Complemented` token for the probe rules processing.

An `OprEvent` node will be removed when its stay in the cache has reached the interval of `NODE_DURATION`. When `RENEW_DURATION` is on, a node's duration will be extended with `NODE_DURATION` from the moment of each update.

When the Event Cache is full, the probe resorts to requesting full messages upon `OprEventChange` objects without the corresponding cached `OprEvent`. It is important not to configure `NODE_DURATION` too large and `MAX_NODES` too small. The longer a node to stay in the cache, the remaining free space will diminish in a higher rate as there is growth of new nodes.

Event Cache data is not written to or read from a persistent file, the data will disappear after the probe shuts down as there is no practical purpose to keep a backup.

Because event correlation and deduplication are determined by `@Identifier` and `@Type` in ObjectServer alerts.status table, which OMi fields to be cached must be consistent with the fields that are used in the probe rules to calculate `@Identifier` and `@Type`.

Constructing the resynchronization event query

To receive resynchronization events from OMi, the probe sends an HTTP request.

The URI in the HTTP request consists of the following parameters: `query`, `start_index`, `page_size`, data ordering flags, and data inclusion flags. These are described in the table that follows.

Table 7. Parameters used in the resynchronization HTTP request

Parameters in the request	**Description**
`query=`	The query expression is configured in the **`ResyncEventQuery`** property.
`start_index=`	This is a running number. The first request's `start_index` is always 1. Each subsequent request's `start_index` is the previous `start_index` plus the `page_size`.
`page_size=`	This value is configured by the **`PageSize`** property. The OMi system receives this number with the first request from the probe and uses this value as the batch size for the query results.

`order_by=`
`order_direction=`
	These parameters allow you to specify which OMi field the resynchronization events are sorted by and whether they are sorted in ascending or descending order. They are configured by the **`ResyncDataOrder`** property using the following format:
	`<field>``:`*`<order_direction>`*, for example: `time_changed:descending`

`include_closed`
`include_relationships`
`include_cis`
`include_history`
`include_annotations`
	These parameters allow you to specify which of the event attributes to include in the resynchronization.
	They are configured by the **`ResyncDataInclusion`** property. For details about how to format the **`ResyncDataInclusion`** property see "Including and ordering data in the resynchronization query" on page 9.

Validating and formatting the resynchronization event query expression

The **`FormatEventQuery`** property controls whether the expression specified in the **`ResyncEventQuery`** property will be validated and formatted for use in the resynchronization request sent to the OMi system.

When **`FormatEventQuery`** is set to `true`, the probe processes the **`ResyncEventQuery`** expression in the following order: validation and formatting.

Each valid query expression can consist of multiple filter query statements joined by the symbols: ^ or | enclosed by parentheses.

Note : ^ represents AND; | represents OR.

Each filter query must comply with the OMi standard.

Validation: The probe checks the syntax of the entire query expression.

Note : The query expression validation process has the following limitations:

- It does not recognize the syntax "`<field> IN (value1, value2,…)`" as valid.
- It does not accept the characters`( ) | ^` in quoted value, for example `id="(abc)"` will be declared an error.

Formatting: The probe converts ^ (caret) and | (pipe) symbols to AND and OR respectively, and encodes the query expression with URL escape characters where necessary.

If **`FormatEventQuery`** is set to `true`, you can write more readable filter expressions without having to manually replace characters with URL escape codes.

The **ResyncEventQuery** property's default value is: `'time_changed GE ++LAST_TIMESTAMP++ ^ time_changed LT ++CURRENT_TIME++'`.

`++LAST_TIMESTAMP++` and `++CURRENT_TIME++` are special tokens recognized by the probe's query expression parser.

In formatting query expressions, the probe encodes certain characters to URL escape codes and replaces ^ with `AND`.

The resultant expression is:

```
time_changed%20GE%202017-11-10T12:34:56.789%2B08:00%20AND%20 time_changed%20LT
%202017-11-16T01:23:45.678%2B08:00
```

The date-time format is: `yyyy-MM-ddThh:mm.ss.SSS+hh:mm`, SSS is millisecond; `+hh:mm` is timezone offset from GMT, for example:

```
http://<omi_host>/opr-web/rest/event_list?query=time_changed%20GE
%202017-11-10T12:34:56.789%2B08:00%20AND%20 time_changed%20LT
%202017-11-16T01:23:45.678%2B08:00& &page_size=10&start_index=1
```

When **ResyncEventQuery** is left blank, the probe will receive all open events from OMi.

If you want to send an event query to OMi without the probe performing syntax validation and URL encoding, set **FormatEventQuery** to `false`. If you do this, you must ensure that the expression is syntactically correct and contains URL escape codes. Take care to escape the `%` character in the property configuration, for example, if the query in URI is:

`“time_changed%20GE%202017-11-10T12:34:56.789%2B08:00”`,

the **ResyncEventQuery** property should be set to:

`“time_changed%%20GE%%202017-11-10T12:34:56.789%%2B08:00”`

Including and ordering data in the resynchronization query

The data to be included in the resynchronization query is specified by the **ResyncDataInclusion** property and the order in which the events are ordered is specified by the **ResyncDataOrder** property.

Selecting data to include in the query

You can use the following parameters to configure the **ResyncDataInclusion** property:

- `include_closed`
- `include_relationships`
- `include_history`
- `include_cis`
- `include_annotations`

You can specify more than one parameter by separating each with a comma.

When a parameter is included in the **ResyncDataInclusion** property, its presence denotes a positive setting. In other words:

`"<parameter>=true"`

When a parameter is excluded from the **ResyncDataInclusion** property, its absence denotes a negative setting. In other words

`"<parameter>=false"`

Whether a parameter appears in the probe's OMi query or not is subject to the parameter's default setting in the OMi system.

If a parameter's OMi default setting is true, when that parameter is not in **ResyncDataInclusion**, then the final query will contain "&<parameter>=false".

If a parameter's OMi default setting is true, when that parameter is in **ResyncDataInclusion**, then the final query will not contain "&<parameter>=true"

If a parameter's OMi default setting is false, when that parameter is not in **ResyncDataInclusion**, then the final query will not contain "&<parameter>=false".

If a parameter's OMi default setting is false, when that parameter is in **ResyncDataInclusion**, then the final query will contain "&<parameter>=true"

Specifying the order of the events in the query

The **ResyncDataOrder** property determines the field by which the data is sorted, and whether it is sorted in ascending or descending order.

Note : For probe rule processing, the incoming data should be sorted by the `time_changed` field in `descending` order.

Initial resynchronization

Event resynchronization often needs a few request iterations to complete.

The number of each iteration is the ceiling value of the quotient of total event record over the configured page size. For example, if the **PageSize** property is set to `10`, and the total number of records reported in the query is 49, then the probe will need five iterations: for the first four requests, each receives a maximum of 10 records; the last request receives 9 records.

If the first resynchronization request fails, then the probe aborts initial resynchronization, writes an error message to the log file, and then exits.

If the first resynchronization request succeeds, but subsequent requests experience a timeout, the probe retries the same request up to the number specified by the **MaxRequestRetry** property. Retry will stop after a batch of events is received without waiting for the timeout period to elapse, or when the maximum number of retries is reached. When an event batch fails to appear, the probe aborts initial resynchronization, writes an error message to the log file, and then exits.

Performing resynchronization from the command interface

You can perform a manual resynchronization using the **resync_default** or **resync_filter** command.

For more information about using the CLI, see "HTTP/HTTPS command interface" on page 11.

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.

Configuring the Probe for HPE OMi for peer-to-peer failover

When configuring the peer-to-peer failover functionality for the Probe for HPE OMi, the host and slave probes must be on different hosts.

You must also configure two connected server entries and two event forwarding rules entries (one of each for the master and the slave hosts). For details about how to configure connected server entries and event forwarding rules entries on HPE OMi, refer to your HPE OMi documentation.

Example property file settings for peer-to-peer failover

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

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

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

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

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

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

Data stream capture

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

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

1. Set the value of the **StreamCapture** property to `true`.
2. Set the value of the **StreamCaptureFile** property to the full path of a directory to hold the data file.

   Notes :

   - Specify the full path to the file. For example:

     `/opt/IBM/tivoli/netcool/omnibus/var/hpe_omi.stream`
   - You cannot include variables such as `$OMNIHOME` in the directory path.
   - The directory must exist. The probe does not create the directory if it does not exist.
3. If the probe is running, restart it.

The probe now writes stream data to the specified file.

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

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, namely `4000`.

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/hpe_omi -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:

  resync_default
  resyn_filter
- *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`.

resync_default

Use the **resync_default** command to perform a resynchronization using the event filter query configured in the **ResyncEventQuery** property.

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

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

The following command performs a resynchronization using the default configuration:

```
$OMNIHOME/bin/nco_http -uri http://<probe_ip>:<http_server_port>/probes/hpe_omi
-datatype application/json -data '{"command":"resync_default","params":[]}' -
method POST
```

resync_filter

Use the **`resync_filter`** command to perform a resynchronization with customized settings.

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

```
-data '{"command":"resync_filter","params":
["event_filter":"<query_expression>", "format_flag":"<boolean_value>"]}'
```

Where:

`event_filter` is the event filter query statement, for example `"time_change>=++LAST_TIMESTAMP+ +"`

`format_flag` is the flag that controls whether the statement is to be validated and formatted in the same way that the query specified by the **`FormatEventQuery`** property can be. For details of how the probe can validate and format the query, see "Constructing the resynchronization event query" on page 7.

`boolean_value` is `true` or `false`

The following command performs a resynchronization taking events that have been updated since the last timestamp, and instructs the probe to perform validation and formatting.

```
$OMNIHOME/bin/nco_http -uri http://<probe_ip>:<http_server_port>/probes/hpe_omi
-datatype application/json -data '{"command":"resync_filter","params":
[{"event_filter":"<query_expression>", "format_flag":"true"}]}' -method POST
```

Note :

The final query statement sent to OMi system will contain other parameters as configured in the following properties: **`ResyncEventQuery`**, **`ResyncDataOrder`**, **`ResyncDataInclusion`**, and **`PageSize`**.

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_default`** command:

```
Information: I-JPR-000-000: RESYNC 'resync_default command received.
Start resynchronization.
```

Storing commands in the nco_http properties file

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

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

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

Data
DataType
Method
URI

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

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

```
Data : '{"command":"resync_default","params":[]}'
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 probe specific properties and command line options configured by the `hpe_omi.props` file.

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

Table 8. Properties and command line options

Property name	**Command line option**	**Description**
Cookie *string*	`-cookie` *string*	Use this property to specify the HTTP cookie name to be retrieved from the probe store. The probe uses the value retrieved from the cookie to replace + *+property_setting++* in the `hpeomiTransport .properties` file. You can specify multiple values for this property by separating each string with a comma (,). The default is `""`. The event source sends the cookie in response to the probe's login request. The default setting for this property instructs the probe to replace the + *+property_setting++* token in the `hpeomiTransport. properties` file with the cookie value.

Table 8. Properties and command line options (continued)

Property name	**Command line option**	**Description**
EnableSSL *string*	`-noenablessl` (This is equivalent to **EnableSSL** with a value of `false`.) `-enablessl` (This is equivalent to **EnableSSL** with a value of `true`.)	Use this property to specify whether SSL connectivity for resynchronization and notification is enabled or disabled. This property takes the following values: `false`: SSL connectivity is disabled. `true`: SSL connectivity is enabled. The default is `false`.
EventCacheConfig *string*	`-eventcacheconfig` *string*	Use this property to specify the configuration file for the Event Cache to store OMi data tokens. The default is `$OMNIHOME/probes/<arch>/ event_cache.cfg`.
FormatEventQuery *string*	`-formateventquery` *string*	Use this property to specify whether the value set in the **ResyncEventQuery** property is subjected to validation or will be used for the OMi query as specified without validation. The default is `true`.
Host *string*	`-host` *string*	Use this property to specify the host name or IP address of the URL used for alert resynchronization. The default is `""`.
KeyStore *string*	`-keystore` *string*	Use this property to specify the location of the keystore file that contains the client certificate for the SSL and trusted authority certificate. The default is `""`.
KeyStorePassword *string*	`-keystorepassword` *string*	Use this property to specify the password required to access the certificate specified by the **KeyStore** property. The default is `""`. **Note :** You can encrypt this password using the `nco_aes_crypt` utility within Netcool/OMNIbus.

Table 8. Properties and command line options (continued)

Property name	Command line option	Description
`MaxRequestRetry` *integer*	`-maxrequestretry` *integer*	Use this property to specify the maximum number of times that the probe retries a resynchronization request if the first resynchronization request was successful. but a subsequent one fails. The default is `10`.
`PageSize` *integer*	`-pagesize` *integer*	Use this property to specify the page size the probe uses for resynchronization data. The default is `10`.
`Password` *string*	`-password` *string*	Use this property to specify the password associated with the **`Username`** property for authentication for the resynchronization request. The default is `""`. **Note :** The probe uses this value to replace the `++Password++` token (if it is specified) in the `hpeomiTransport.properties` file.
`Port` *integer*	`-port` *integer*	Use this property to specify the host port that OMi uses for alert notification. The default is `0`.
`ResyncDataInclusion` *string*	`-resyncdatainclusion` *string*	Use this property to specify the data inclusion query parameters. This property accepts the following values: • `include_annotations` • `include_cis` • `include_closed` • `include_history` • `include_relationships` The default is `include_cis, include_relationships`. **Note :** For details about configuring this property, see “Constructing the resynchronization event query” on page 7 and “Including and ordering data in the resynchronization query” on page 9.

Table 8. Properties and command line options (continued)

Property name	**Command line option**	**Description**
ResyncDataOrder *string*	-resyncdataorder *string*	Use this property to specify the field and direction for resynchronization data order. The default is time_changed:ascending.
ResyncEventQuery *string*	-resynceventquery *integer*	Use this property to specify the query the probe sends to resynchronize events from OMi. The default is time_changed GE + +LAST_TIMESTAMP++ ^ time_changed LT ++CURRENT_TIME ++.
ResyncTimeout *integer*	-resynctimeout *integer*	Use this property to specify the time (in seconds) that the probe allows for a resynchronization attempt. The default is 30.
StreamCapture *string*	-streamcapture *string*	Use this property to specify whether or not the probe stores the event data in a stream capture file. The default is false.
StreamCaptureFile *string*	-streamcapturefile *string*	Use this property to specify the location of the stream capture file. The default is ${OMNIHOME}/var/ hpe_omi.stream. On UNIX, if you specify an environment variable (for example, OMNIHOME) you must include it within curly brackets { }. On Windows operating systems, you must manually change this property value to: %OMNIHOME%\\var\\ hpe_omi.stream.

Table 8. Properties and command line options (continued)

Property name	Command line option	Description
`TransportFile` *string*	`-transportfile` *string*	Use this property to specify the location of the transport properties file. The default is `${OMNIHOME}/java/conf/hpeomiTransport.properties` On UNIX, if you specify an environment variable (for example, `OMNIHOME`) you must include it within curly brackets `{ }`. On Windows operating systems, you must manually change this property value to: `%OMNIHOME%\\java\\conf\\ hpeomiTransport.properties`
`TransportType` *string*	`-transporttype` *string*	Use this property to specify the transport method. The default is `Webhook`. **Note :** Currently `Webhook` is the only supported value for this property.
`UseLastTimestamp` *string*	`-uselasttimestamp` *string*	Use this property to specify whether the probe uses the last event timestamp to replace the `++LAST_TIMESTAMP++` token if configured in **`ResyncEventQuery`** property. See "Constructing the resynchronization event query" on page 7. `true`: The probe replaces the `++LAST_TIMESTAMP++` token with the last recorded event timestamp. If event timestamp is not available, then the `++LAST_TIMESTAMP++` token will be the time value 24 hours before the current time. `false`: The probe replaces `++LAST_TIMESTAMP++` with the time value 24 hours before the current time. The default is `true`.
`Username` *string*	`-username` *string*	Use this property to specify the username used in the authentication for the resynchronization request. The default is `""`. **Note :** The probe uses this value to replace the `++Username++` token (if it is specified) in the `hpeomiTransport.properties` file.

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 9. Properties and command line options

Property name	Command line option	Description
`CommandPort` *integer*	`-commandport` *integer*	Use this property to specify the port to which users can Telnet to communicate with the probe using the Command Line Interface (CLI) supplied. 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`.
`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).

Table 9. Properties and command line options (continued)

Property name	Command line option	Description
`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 9. 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`.

OMi event attributes

The set of attributes that the probe receives from OMi events depends on whether the events are resynchronization events or notification events. They can be different.

The availability of certain event attributes depends on the following factors:

- The data inclusion parameters used in the resynchronization query.
- The data in the OMi event, for example if an OMi event is never annotated, its OprEvent contains no annotation attributes. In notification, each forwarded OprEvent carries all attributes available in the event.

The following table describes which event attributes are included for each of the resynchronization inclusion parameters.

Table 10. Attributes available for each inclusion parameter

Data inclusion setting	**Event attributes available**
`include_annotation=true`	`$(annotation_list.*)`
`include_history=true`	`$(history_line_list_ref.*)`

`include_cis=true`	`$(related_ci.configuration_item.*)`
	 `$(source_ci. configuration_item.*)`
	 Note : When `include_cis=false`, the minimal
	 `$(*.configuration_item.*)`
	 tokens are:
	 • `*.configuration_item.id`
	 • `*.configuration_item.type`
	 • `*.configuration_item.type_label`

`include_relationships=true`	`$(related_ci.configuration_item.part_of.*)`
	 `$(source_ci.configuration_item.part_of.*)`

Processing considerations

Certain event tokens are indexed by the probe for the convenience of enumerating and safe referencing in the probe rules. Indexed tokens apply to the following attributes:

- `annotation_list`
- `forwarding_info_list`
- `history_line_list_ref`

The range of each event token index is from `0` to `(<attribute>.count-1)`.

The bigger the set of event attributes you include, the bigger the overhead that is incurred and the greater the memory usage in HTTP transactions and OprEvent data extraction. To make the probe work efficiently, include only the attributes relevant to production.

Writing probe rules

When customizing probe rules, you must take care when selecting the fields for the Netcool event's `@Identifier` and `@Type`, the changes made must be able to facilitate event correlation and deduplication in the ObjectServer.

`OprEvent`'s ID field is the OMi event's alert key.

`OprEventChange`'s ID field is the message ID to the OMi event update, `event_ref.target_id` is the OMi origin event's alert key. `OprEventChange`'s id cannot be used in `@Identifier`, otherwise event correlation will fail.

Although every `OprEvent`'s ID field is unique, the ID field alone is sufficient to be Netcool event `@Identifier`, however it is a good practice to combine the ID field with other OMi event attributes to form `@Identifier` to create a virtual boundary that helps avoid identifier clash, because the ObjectServer might be receiving events from other target systems.

OMi event attribute tokens in the probe rules

For a full list of all possible OMi event attribute tokens that can be used with the probe rules file, see the following page on the Knowledge Center:

https://www.ibm.com/support/knowledgecenter/SSSHTQ/omnibus/probes/hpe_operations_manager_i/wip/reference/hpe_omi_elements_attr_tokens_list.html

Error messages

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

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

Table 11. Error messages

Error	**Description**	**Action**
`Fail to start Transport module for connection`	The probe failed to connect to the OMi server.	Check that you have configured the `hpeomiTransport.properties` file correctly. See "Configuring the transport module" on page 5.
`Fail to get active alarms during resync`	The probe's alarm resynchronization request failed.	Check that you have configured the resynchronization query correctly. See "Constructing the resynchronization event query" on page 7.

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

Table 12. ProbeWatch messages

ProbeWatch message	**Description**	**Triggers/causes**
`Connection to the event source lost`	This ProbeWatch message reports that the connection to the event source was lost.	The probe lost its connection to the event source.

Table 12. ProbeWatch messages (continued)

ProbeWatch message	**Description**	**Triggers/causes**
`Start resynchronization`	The probe started the resynchronization process.	A resynchronization request was sent to the OMi.
`Finish resynchronization`	The probe finished resynchronization process.	OMi sent the final resynchronization event to the probe .
`Resynchronization fails`	This ProbeWatch message reports a failure of the resynchronization process.	The resynchronization process failed.

Appendix A. Notices and Trademarks

This appendix contains the following sections:

- Notices
- Trademarks

Notices

Trademarks

SC27-8772-00