IBM® Tivoli® Netcool/OMNIbus SNMP Writer Gateway
8.0

Reference Guide
October 30, 2020

SC23-7804-10

Notice

Before using this information and the product it supports, read the information in Appendix A, “Notices and Trademarks,” on page 33.

Edition notice

This edition (SC23-7804-10) applies to version 8.0 of IBM Tivoli Netcool/OMNIbus SNMP Writer Gateway and to all subsequent releases and modifications until otherwise indicated in new editions.

This edition replaces SC23-7804-09.

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 SNMP Writer Gateway documentation is provided in softcopy format only. To obtain the most recent version, visit the IBM® Tivoli® Knowledge Center:

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

Table 1. Document modification history

Document version	Publication date	Comments
SC23-7804-00	February 20, 2006	First IBM publication.
SC23-7804-01	June 20, 2007	Support added for the AES encryption of string values that can be assigned to attributes within the WRITER section of the gateway configuration file. The SNMP Writer Gateway no longer supports electronic licensing. The gateway now uses the IBM® software licensing process. Support for SNMP version 2 traps added.
SC23-7804-02	December 19, 2007	Support for IBM Tivoli® Netcool®/OMNIbus™ V7.2 added. Sybase support information added.
SC23-7804-03	March 28, 2008	Summary section updated. Running the ObjectServer in a secure mode section added.
SC23-7804-04	September 11, 2009	Guide rewritten to reflect updated gateway architecture.

Table 1. Document modification history (continued)

Document version	Publication date	Comments
SC23-7804-05	February 25, 2011	Updated "Summary" on page 1. "Properties and command line options" on page 18 updated to describe the following properties: • **Gate.Mapper.Debug** • **Gate.Mapper.ForwardHistoricDetails** • **Gate.Mapper.ForwardHistoricJournals** • **Gate.Reader.Debug** • **Gate.Reader.Description** • **Gate.Reader.DetailsTableName** • **Gate.Reader.FailbackEnabled** • **Gate.Reader.FailbackTimeout** • **Gate.Reader.IducFlushRate** • **Gate.Reader.JournalTableName** • **Gate.Reader.LogOSSql** • **Gate.Reader.Password** • **Gate.Reader.ReconnectTimeout** • **Gate.Reader.Server** • **Gate.Reader.StatusTableName** • **Gate.Reader.Username** SNMP trap types described in "Specifying values for the generic and specific fields in SNMP traps" on page 14. Event flush rate described in "Controlling the frequency at which the gateway requests updates from the ObjectServer" on page 16. Updated "Store-and-forward mode" on page 9. Updated "Running the gateway" on page 29. Updated "Error messages" on page 30.
SC23-7804-06	August 3, 2012	Summary section updated.
SC23-7804-07	November 30, 2012	Guide updated for Netcool/OMNIbus V7.4 release. "Installing the gateway" on page 3 updated.
SC23-7804-08	November 8, 2013	"Summary" on page 1 updated. "Startup command file" on page 17 updated. "Table replication definition file" on page 7 updated. "FIPS mode and encryption" on page 9 added to the guide.

Table 1. Document modification history (continued)		
Document version	**Publication date**	**Comments**
SC23-7804-09	July 28, 2016	The guide has been reorganized. As part of this reorganization, the following new topics have been added: • "Overview" on page 2 • "Features of the IBM Tivoli Netcool/OMNIbus SNMP Writer Gateway" on page 3 • "Configuring the gateway" on page 5 • "Authentication" on page 6 • Enabling the gateway "Enabling the gateway to read alerts from the ObjectServer" on page 7 • "Gateway operation" on page 12 "Summary" on page 1 updated. "Properties and command line options" on page 18 updated to describe the following new property: • **Gate.SNMP.LocalName** Addresses the following enhancement requests: • RFE 20320: Forwarding configurable IP Address within SNMP trap • RFE 76266: Gateway to log PDU Request ID values
SC23-7804-10	October 30, 2020	Updated for version 8 of the gateway. "Summary" on page 1 updated. Description for **Gate.SNMP.RetryInterval** added to, and descriptions for **SecurityAuthProtocol** and **SecurityPrivProtocol** updated in, "Properties and command line options" on page 18. Version 8 also addresses the following APARs: • **IJ21440**: Fixed issue which made probe not able to receive traps with AES, AES-192 and AES-256 privacy types. • **IJ28244**: No delay for next retry attempt to send trap after previous send fails.

Conventions used in this guide

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

Operating system-dependent variables and paths

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

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

```
$OMNIHOME/gates
```

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

```
%OMNIHOME%\gates
```

For gateways supported on UNIX, Linux, and Windows operating systems, gateway guides use the standard UNIX conventions for specifying environment variables and describing directory paths. When using the Windows command line with these gateways, 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.

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/platform/arch
```

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

Note : This gateway 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

Gateways 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. SNMP Writer Gateway

The IBM Tivoli Netcool/OMNIbus SNMP Writer Gateway forwards Netcool/OMNIbus® alerts as Simple Network Management Protocol (SNMP) traps to an SNMP reader, such as the IBM Tivoli Netcool/OMNIbus SNMP Probe. This allows Tivoli Netcool/OMNIbus to generate traps that are forwarded to another management platform (referred to as an SNMP writer application) such as SunNet Manager or HP Network Node Manager.

The SNMP Writer Gateway supports versions 1, 2, and 3 of the Simple Network Management Protocol.

This guide contains the following sections:

Summary

Use this summary information to learn about the SNMP Writer Gateway.

The following table provides a summary of the gateway:

Table 3. Summary	
Gateway target	Simple Network Management Protocol agent
Gateway executable file name	`nco_g_snmp` (on UNIX and Linux operating systems) `nco_g_snmp.exe` (on Windows operating systems)
Gateway installation package	`omnibus-`*`arch`*`-gateway-nco_g_snmp-`*`version`*
Package version	8.0
Gateway 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=swg21613133

Table 3. Summary (continued)	

Configuration files	Map definition file:
	`$OMNIHOME/gates/snmp/snmp.map`
	Properties file:
	`$OMNIHOME/gates/snmp/NCO_GATE.props`
	Table replication definition file:
	`$OMNIHOME/gates/snmp/snmp.reader.tblrep.def`
	Startup command file:
	`$OMNIHOME/gates/snmp/snmp.startup.cmd`

Additional binaries	None
Requirements	A currently supported version of IBM Tivoli Netcool/OMNIbus `common-libncrypt-1_0` (UNIX)
Remote connectivity	Available
Failover/failback functionality	Available
Multicultural support	Available For information about configuring multicultural support, including language options, see the *IBM Tivoli Netcool/OMNIbus Installation and Deployment Guide*.
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*.

Overview

The IBM Tivoli Netcool/OMNIbus SNMP Writer Gateway is a uni-directional gateway, that creates and forward traps based on alerts present in the Object Server.

It offers flexible event filtering that allows the user to specify which alerts the gateway uses to create traps.

Features of the IBM Tivoli Netcool/OMNIbus SNMP Writer Gateway

The IBM Tivoli Netcool/OMNIbus SNMP Writer Gateway enables users to map table data in order to generate and forward traps.

Trap forwarding

The gateway uses the Insert, Delete, Update, or Control (IDUC) communication protocol to retrieve events from ObjectServer tables (specifically, alerts.status). The gateway can replicate the data from those tables to the destination server.

Details of the tables to be replicated are stored in the table replication definition file. The events retrieved from these tables are based on the table replication definition file configuration, including their filtering. Retrieved events are then passed through a mapper to assign values to target fields will be used to create a trap. The mappers are specified in the table replication definition file and defined in the map definition file.

Mapping table data

The gateway writes the alerts received from the various tables in the ObjectServer into traps. The mapping is done by the mapper defined in the snmp.map map definition file.

The gateway can forward updates on alerts.

Installing the gateway

There are separate procedures for installing the gateway on each version of Tivoli Netcool/OMNIbus.

Follow the procedure for the version of Tivoli Netcool/OMNIbus that your site uses.

Installing probes and gateways on Tivoli Netcool/OMNIbus V8.1

From Tivoli Netcool/OMNIbus V8.1 onwards, Tivoli Netcool/OMNIbus probes and gateways can be installed using the IBM Installation Manager. One of the key features of Installation Manager is that all platforms are shipped in a single ZIP file, which means that you do not have to select the platform that you require; Installation Manager does it for you.

Before you can install a probe or gateway, you must have installed and configured Installation Manager and Tivoli Netcool/OMNIbus. To install probes and gateways, you must make sure that the Core Tivoli Netcool/OMNIbus features **Probe Support** and **Gateway Support** respectively are installed.

Installing probes and gateways using the Command Line Tool

To install the probe or gateway using the Command Line Tool, run the following command:

```
installation_manager_location/eclipse/tools/imcl -c install
com.ibm.tivoli.omnibus.integrations.integration_name -repositories
repository_containing_required_integration -installationDirectory
location_of_netcool_omnibus_install_you_are_installing_into
```

Where *integration_name* specifies the name of the probe or gateway that you want to install.

You will be prompted to agree to the terms and conditions of the license as a prerequisite for installing the integration. If you have already reviewed the license and want to skip the manual acceptance, add the `-acceptLicense` option to the `install` command to silently agree to the license.

The following is an example command used to install the SNMP Probe:

```
imcl -c install com.ibm.tivoli.omnibus.integrations.nco-p-mttrapd -
repositories /home/my_home_dir/nco-p-mttrapd_im_package -
installationDirecory /opt/IBM/tivoli/netcool
```

Where /home/my_home_dir/nco-p-mttrapd_im_package contains the unzipped contents of the SNMP Probe Installation Manager package.

Note : The command line tool does not add the repository permanently to the Installation Manager instance. If you subsequently start the Installation Manager GUI, the repositories will not be present in the **Repositories** dialog box.

Uninstalling probes and gateways using the Command Line Tool

To uninstall the probe or gateway using the Command Line Tool, run the following command:

installation_manager_location/eclipse/tools/imcl uninstall com.ibm.tivoli.omnibus.integrations.*integration_name* -installationDirectory *location_of_netcool_omnibus_install_you_are_uninstalling_from*

Where *integration_name* specifies the name of the probe or gateway that you want to uninstall.

The following is an example command used to uninstall the SNMP Probe:

```
imcl uninstall com.ibm.tivoli.omnibus.integrations.nco-p-mttrapd -
installationDirecory /opt/IBM/tivoli/netcool
```

Installing probes and gateways using the GUI

To install the probe or gateway using the GUI, use the following steps:

1. Unzip the IM package that contains the probe or gateway into a directory of your choosing. A file called repository.config will appear after unzipping the IM package.
2. Start Installation Manager using the following command:

   installer_path/IBMIM

   Where *installer_path* is the path to the Installation Manager directory.
3. Perform the following menu actions to display the repository dialog box:

   Files > **Preferences** > **Repositories.**
4. Use the button **Add Repository** in the repository dialog box to point to the repository that contains the unzipped IM package that contains the probe or gateway. This is the repository that contains the repository.config file.
5. Click the **Install software packages** icon.
6. Select the name of the probe or gateway that you want to install.
7. Click **Next**.
8. Click **I accept** when the Licensing panel appears.
9. Highlight **IBM Tivoli Netcool OMNIbus** in the **Package Group Name** field.
10. Click **Next**.
11. Click **Next**.
12. Click **Install**.
13. When the **Install Packages** panel appears indicating that you have successfully installed the probe or gateway, click **Finish**.

Uninstalling probes and gateways using the GUI

To uninstall the probe or gateway, use the following steps:

1. Start Installation Manager using the following command:

   installer_path/IBMIM

   Where *installer_path* is the path to the Installation Manager directory.

2. Click the **Uninstall software packages** icon.
3. Select the name of the probe or gateway that you want to uninstall.
4. Click **Next**.
5. Click **Uninstall**.
6. When the **Install Packages** panel appears indicating that you have successfully uninstalled the probe or gateway, click **Finish**.

Configuring the gateway

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

Table 4 on page 5 lists gateway configuration tasks. For each configuration task, the table lists the properties you use with that task, and the section in this guide that shows you how to complete the configuration task.

Some configuration tasks are mandatory for all installations. For those configuration tasks set the properties to the correct values or verify that their default values are suitable for your environment. The remaining configuration tasks are optional depending on which ones you want to use.

Note : The table references the following categories of properties that are defined in the `NCO_GATE.props` properties file:

- Common gateway properties
- SNMP Gateway Properties

For reference information and command line options on these properties, see "Properties and command line options" on page 18.

Table 4. Configuring the gateway

Configuration tasks	Properties	See
Required configuration tasks:		
Creating the gateway server		
Create the gateway server in the Tivoli Netcool/OMNIbus interfaces file	**None**	"Configuring the gateway server" on page 6
Authentication configuration task		
Authenticate the gateway with the ObjectServer. **Note :** The gateway needs to authenticate with the ObjectServer only when the ObjectServer runs in secure mode.	**`Gate.Reader.Password`** **`Gate.Reader.Username`**	"Authentication" on page 6
Connection Configuration Task		
Enabling the gateway to read alerts from the ObjectServer	**`Gate.Reader.Server`**	"Enabling the gateway to read alerts from the ObjectServer" on page 7
Optional configuration tasks:		
Table replication configuration task		

Table 4. Configuring the gateway (continued)		
Configuration tasks	**Properties**	**See**
Define the tables and event types that are replicated between the ObjectServer and the SNMP writer.	**Gate.Reader.TblReplicateDefFile**	"Table replication definition file" on page 7
Data mapping configuration task		
Define how fields in ObjectServer tables map to SNMP PDU variable bindings.	**Gate.Mapfile**	"Map definition file" on page 8
FIPS mode and encryption configuration tasks (for FIPS support for property encryption between the ObjectServer and the gateway)		
Operate the gateway using FIPS mode and encryption.	**None**	"FIPS mode and encryption" on page 9
Operate the gateway using AES encryption.	**None**	"AES encryption" on page 10

Configuring the gateway server

You must create the gateway server in the Tivoli Netcool/OMNIbus interfaces file.

Use the Server Editor (`nco_xigen`) to specify the host name and port number of the gateway server in the Tivoli Netcool/OMNIbus interfaces file.

You must make the gateway server name the same as the name of the gateway properties file. For example, the default properties file supplied with the gateway is called `NCO_GATE.props`, so the corresponding gateway server name is `NCO_GATE`.

The **Name** property value is incorporated into the file name used to locate the property file. For example, if the default property file location is used, the gateway will look for a file `$OMNIHOME/etc/<Name>.props` where `$OMNIHOME/etc` is the default property file's directory and `<Name>` is the value specified by the `Name` property.

Note : For information about using the Server Editor, see the *IBM Tivoli Netcool/OMNIbus Administration Guide*.

Authentication

The gateway needs to authenticate itself with the ObjectServer only when the ObjectServer runs in secure mode.

Authenticating the gateway with the ObjectServer

When the ObjectServer runs in secure mode, it requires each gateway that connects to it to supply a user name and password. Set the gateway's **Gate.Reader.Username** and **Gate.Reader.Password** to the user name and password of the gateway's ObjectServer account.

Note : A gateway user account needs to be created within Tivoli Netcool/OMNIbus so that the gateway can supply the username and password to identify itself to the ObjectServer.

Enabling the gateway to read alerts from the ObjectServer

To enable the gateway to read alerts from the ObjectServer, use the **Gate.Reader.Server** property.

The **Gate.Reader.Server** property specifies the name of the ObjectServer from which the gateway reads alerts. The name can either be an interface name (for example, NCOMS) or the <host>:<port> details of the ObjectServer.

Table replication definition file

The gateway replicates data between ObjectServer tables and the gateway target. The table replication definition file is used to define which tables and event types are monitored in Tivoli Netcool/OMNIbus and forwarded to the target that the gateway is configured to send data to.

You can specify the location of the table replication definition file using following generic Tivoli Netcool/OMNIbus property.

Gate.Reader.TblReplicateDefFile

The default table replication definition file is in the following directory: $OMNIHOME/gates/snmp/snmp.reader.tblrep.def

The default table replication definition file contains example commands. You should make a backup copy of the default file for future reference.

Note : You should use the REPLICATE command to replicate data from the primary table (alerts.status) and dynamic secondary tables (if required).

You can add one or more optional clauses to the REPLICATE command to further process the data during replication. The available commands are listed in the following syntax example. Use the optional clauses in the order in which they are listed in the syntax. For example, when using both the

FILTER WITH and AFTER IDUC DO clauses, the FILTER WITH clause must precede the AFTER IDUC DO clause.

```
REPLICATE ALL | (INSERTS, UPDATES, DELETES)
FROM TABLE sourcetable
USING MAP mapname
[FILTER WITH filter]
[INTO targettable]
[ORDER BY order, ... ]
[AFTER IDUC DO afteriduc] ;
```

Table 5. Optional replication commands

Command	Description
FILTER WITH '*filter*'	Filters the database rows selected for replication, where *filter* defines the filter that the gateway uses in the WHERE clause of the SQL SELECT. Filtering is positive by default, which means that only those events that match the filter definition are replicated. You can use a negative filter by putting an exclamation mark (!) before the equals sign (=) in the filter clause. For example, the following filter clause replicates all events whose severity is not 5: FILTER WITH 'Severity !=5'

Table 5. Optional replication commands (continued)	
Command	**Description**
`ORDER BY 'order'`	Order results by the `SQL SELECT ORDER BY` clause used to get data. A potential use case might be to order by first occurrence, so that alerts are processed in chronological order, in which case the value specified for *order* would be `'FirstOccurrence'`.
`AFTER IDUC DO 'afteriduc'`	Updates replicated rows, where *afteriduc* specifies which field to update with what value. This uses the `SQL UPDATE` action to execute on rows retrieved by the `SQL SELECT` action used to get data, e.g. `'SentToCRM=1'`.

Map definition file

The map definition file defines how the gateway maps data received from the OS tables into SNMP traps.

The default map definition file is `$OMNIHOME/gates/snmp/snmp.map`.

Syntax

You can configure the mapping functions of the gateway by using the mapper attributes.

Mappings for use with the SNMP Writer Gateway must use the following syntax:

```
CREATE MAPPING mappingname
(
'varbindint' = '@fieldname',
['varbindint' = '@fieldname' [ ON INSERT ONLY ], ]...
) ;
```

where:

- *mappingname* is the name of the mapping to be created.
- *varbindint* is the integer value for the `varbind` field in the SNMP trap.
- *fieldname* is the name of a field in the ObjectServer `alerts.status` table.

Example mapping

The following default mapping is supplied in the `snmp.map` file:

```
CREATE MAPPING StatusMap
(
        '0' = '@Summary',
        '1' = '@Severity',
        '2' = '@Location',
        '3' = '@Node',
        '4' = '@AlertGroup',
        'Node' = '@Node'
);
```

Defining a lookup table

You can define a lookup table within the map file and use it to transform values found in particular fields to a different set of values before constructing SNMP traps.

The following is an example lookup table definition:

```
CREATE LOOKUP ExampleTable (
        { 'source1', 'target1'},
```

```
        { 'source2', 'target2'}
) DEFAULT = '' ;
```

This example is used in the map as follows:

It is important to convert @myfield to be compatible with its respective CREATE LOOKUP datatype.

```
'3' = Lookup('@myfield', 'ExampleTable')
```

For example:

```
CREATE  LOOKUP SeverityTable (
        {'0','Clear'},
        {'1','Indeterminate'},
        {'2','Warning'},
        {'3','Minor'},
        {'4','Major'},
        {'5','Critical'}
) DEFAULT = 'Unknown' ;

CREATE MAPPING StatusMap
(
        '0' = '@Summary',
        '1' = Lookup(TO_STRING('@Severity'), 'SeverityTable'),
        '2' = '@Location',
        '3' = '@Node',
        '4' = '@AlertGroup',
        'Node' = '@Node'
)
```

Note : Changes or additions to the lookup table will only take effect after the gateway is stopped and restarted.

Store-and-forward mode

Store and forward mode functionality is no longer supported in this version of the gateway.

FIPS mode and encryption

This gateway complies with Federal Information Processing Standard 140-2 (FIPS 140-2). It can be run in FIPS mode on any currently supported version of Tivoli Netcool/OMNIbus.

You can use encryption algorithms to secure string value entries made in the properties file, including passwords. You must use the generic Tivoli Netcool/OMNIbus **ConfigCryptoAlg** property to specify the encryption method and the generic Tivoli Netcool/OMNIbus **ConfigKeyFile** property to specify the encryption key file, amongst a number of other required settings.

For more information about running the gateway in FIPS mode, and encrypting properties and passwords, see *Running the ObjectServer in secure mode*, *Running the proxy server in secure mode*, and *Encrypting plain text passwords in routing definitions* in the *IBM Tivoli Netcool/OMNIbus Administration Guide*.

Also see, *Configuring FIPS 140-2 support for the server components* in the *IBM Tivoli Netcool/OMNIbus Installation and Deployment Guide*.

Also see *SSL and FIPS 140-2 support* in the *IBM Tivoli Netcool/OMNIbus Event Integration Facility Reference*.

Also see *Appendix C. WAAPI security* in the *IBM Tivoli Netcool/OMNIbus Web GUI Administration API (WAAPI) User's Guide*.

Note : If you run the gateway in FIPS mode, you must either use no encryption, or if you do use encryption, you must use `nco_aes_crypt` with the cipher (`-c`) option `AES_FIPS`. The cipher option used here must match the option specified by the **ConfigCryptoAlg** property. For example:

`$NCHOME/omnibus/bin/nco_aes_crypt -c AES_FIPS -k` *`key_file string_value`*

AES encryption

AES encryption can be used to encrypt any string within the gateway writer section of the configuration file. It is used by the gateway to prevent sensitive data from being available in readable format in the gateway configuration file.

Note : AES encryption is supported on Tivoli Netcool/OMNIbus V7.4.0 (and above) on all operating systems.

nco_aes_crypt

You can encrypt strings in the gateway configuration file using the `nco_aes_crypt` tool (supplied with Tivoli Netcool/OMNIbus). The syntax of encrypted data is as follows:

```
@datalength:encrypted_data@
```

Where *datalength* is the length of the data in bytes (expressed as a decimal) and the data itself is base64 encoded. The at sign (`@`) indicates the start and end of the encrypted data definition. The colon (`:`) acts as a field separator.

The encrypted values appear in single quotes on the right side of expressions in the writer section of the configuration file. The following is an example line from a configuration file showing the host name given in encrypted format:

```
HOST ='@64:lHBLuIPLNye8zCWhykFVFY7y90V9kCjGK5GSWu5VBdSlgQOqarq6T4UK4xk5Vqix@'
```

Note : You can obtain the `nco_aes_crypt` tool from the IBM Passport Advantage website: http://www-306.ibm.com/software/howtobuy/passportadvantage/pao_customers.htm. Access the Software Downloads section and search for `Netcool/OMNIbus Gateway configuration encryption library`.

Using the nco_aes_crypt tool

Data in the configuration file must be encrypted using the `nco_aes_crypt` tool.

This is a command line tool which takes the following format:

```
nco_aes_crypt [-d | -e] [-o outfile] -k keyfile -f filename
nco_aes_crypt [-d | -e] [-o outfile] -k keyfile data
```

The output of this command will be the encrypted string to be used in the configuration file.

The following table describes the options available with `nco_aes_crypt`:

Table 6. nco_aes_crypt command line options

Command line option	Description
`-d` or `-e`	Use this option to specify the mode in which the `nco_aes_crypt` tool runs: d - decrypt mode e - encrypt mode The default is e.
`-o` *string*	Use this option to specify the output file to which the encrypted data will be written.
`-k` *string*	Use this option to specify the path of the file containing the key data.

Table 6. nco_aes_crypt command line options (continued)	
Command line option	**Description**
`-f` *string*	Use this option to specify the path of the file containing data requiring encryption.
data	Use this option to specify the data to be encrypted or decrypted.

Encryption key file

The encryption key is stored in a flat file alongside the encrypted data. The key storage file has an ASCII numeric key length indicator followed by a colon and the key in binary form.

The format of the key file is as follows:

```
key_length:key_data
```

Where `key_length` is the length of the key in bits and the `key_data` is the key in binary form. Valid length values are 128, 192 and 256.

For example:

```
128:1234567812345678
```

In this case, `key_length` is 128 since the ASCII string 1234567812345678 has 16 bytes (128 bits).

You can generate random or pre-defined keys of varying lengths using `nco_keygen`. To generate a key file, use the following command:

```
nco_keygen -o outfile[-l length|-k]key][-h |-?]
```

The following table gives the descriptions of the above command line options.

Table 7. Encrytion key file command line options	
Command line option	**Description**
`-o` *outfile*	Use this option to specify the output file name.
`-l` *length*	Use this option to specify the length (in bits) of the key to write out. The default is 128. **Note :** The value that you specify must be divisible by 8.
`-k` *key*	Use this option to specify the key to be written out, expressed as hex digits. **Note :** This option bypasses automatic key generation.
`-h` /*-?*	Use this option to print the help information and exit

Note : AES encryption is used as the initial encryption method for sensitive data. However, this does not mean that the data can be considered to be secure purely due to AES encryption; the security of the data depends on the restriction of access to the key file used for AES encryption. Access to this file is controlled using UNIX file permissions.

Using encrypted data

To use encrypted data, you must add the following line to the WRITER section of the configuration file:

```
LOAD ENCRYPTION KEY FROM 'key_file_path' USING 'AES'
```

Where *key_file_path* is the path to the file containing the encryption key.

Running the ObjectServer in a secure mode

When the gateway connects to the ObjectServer running in secure mode, it needs to authenticate with a user name and password. This user name and password can be encrypted using the `nco_aes_crypt` tool.

To enable the encryption, the location of the key file must be specified at the beginning of the configuration file. This is followed by the AUTH_USER and AUTH_PASSWORD fields which contain the encrypted user name and password required for authentication.

The following example shows the three fields that need to be added at the beginning of the configuration file when the ObjectServer runs in a secure mode:

```
LOAD ENCRYPTION KEY FROM '/HOME/72/solaris/omnibus/keyflie_name' USING
'AES';
AUTH_USER '@44:2yXgd6fp9q1Ey4sSAb2RibzA3+PpCZmhAZXo6nNdkvQ=@'; #
encrypted_user_name
AUTH_PASSWORD '@44:mdyEb8VTh+2wALnNlR7dnGnxRZ3BkMOQbR5IgxLlHuc=@'; #
encrypted_password
```

Gateway operation

Use this information to learn how the SNMP Writer Gateway interacts with the Simple Network Management Protocol (SNMP). This includes information on how the gateway interacts with SNMP with regard to SNMP generic and specific traps and forwards the traps to an SNMP agent (gateway target).

Table 8 on page 12 identifies some of the operations related to how the gateway interacts with SNMP, including those related to SNMP generic and specific traps operations. For each operation, the table lists the properties you use to control how the gateway performs the operation, and the section of the guide that provides details about the operation and the valid values for the associated properties. For each operation, set the properties to the correct values or verify that their default values are suitable for your environment.

Note : The table references the following categories of properties that are defined in the `NCO_GATE.props` properties file:

Common Gateway Properties
SNMP Gateway Properties

For reference information and command line options on these properties, see "Properties and command line options" on page 18.

Table 8. Configuring gateway operation

Configure gateway operation tasks	Properties	See
Running the gateway using the TCP transport protocol	**Gate.SNMP.Gateway** **Gate.SNMP.Protocol**	"Running the gateway using the TCP transport protocol" on page 13
Specifying values for the generic and specific fields in SNMP traps	**Gate.SNMP.Specific** **Gate.SNMP.Trap**	"Specifying values for the generic and specific fields in SNMP traps" on page 14

Table 8. Configuring gateway operation (continued)		
Configure gateway operation tasks	**Properties**	**See**
Specifying SNMPv3 support	**Gate.SNMP.EngineID** **Gate.SNMP.SecurityName** **Gate.SNMP.SecurityLevel** **Gate.SNMP.Security AuthProtocol** **Gate.SNMP.Security AuthPassphrase** **Gate.SNMP.Security PrivProtocol** **Gate.SNMP.Security PrivPassphrase**	"Specifying SNMPv3 support" on page 15
Controlling the frequency at which the gateway requests updates from the ObjectServer	**Gate.Reader.IducFlushRate**	"Controlling the frequency at which the gateway requests updates from the ObjectServer" on page 16
Forwarding alert updates	**Gate.SNMP.ForwardUpdates**	"Forwarding alert updates" on page 16
Controlling the size of the hash table cache	**Gate.CacheHashTblSize**	"Controlling the size of the hash table cache" on page 16
Specifying the gateway to run under Process Agent control	**Gate.PAAware**	"Specifying the gateway to run under Process Agent control" on page 17
Controlling forwarded TCP/IP Source IP Address	**Gate.SNMP.LocalName**	"Controlling the size of the hash table cache" on page 16
Controlling SNMP Agent Address by Node entry	There is no associated property for the node value. The node value is specified within the map file.	"Controlling SNMP Agent Address by Node entry" on page 17

Running the gateway using the TCP transport protocol

You can run the gateway using the Transmission Control Protocol (TCP) transport protocol.

To run the gateway using TCP, set the **Gate.SNMP.Protocol** property to TCP and specify an IP address and port number for the **Gate.SNMP.Gateway** property.

For example:

```
Gate.SNMP.Gateway     : '9.20.1.1:162'
Gate.SNMP.Protocol : 'TCP'
```

When the gateway is using TCP, it sends traps to the specified port and it requires an application (for example, the SNMP Probe) to be listening on that port. If the listening application shuts down, the gateway logs an error message and attempts to restart the connection.

Note : Store-and-forward mode requires the gateway to use the TCP protocol. See “Store-and-forward mode” on page 9 for more information.

Specifying values for the generic and specific fields in SNMP traps

The SNMP standard defines seven trap types that are generated by SNMPv1 agents: six generic traps and one enterprise-specific trap. The enterprise-specific trap is used by organizations to define traps for particular devices. The generic trap types are fixed and cannot be changed, whereas it is possible to define multiple enterprise-specific traps.

You can use the **Gate.SNMP.Trap** property to specify an integer value for the generic trap field in SNMP traps.

Table 9 on page 14 lists the six generic trap types defined for SNMPv1 agents.

Table 9. Generic SNMP trap types

Trap type	Integer value
coldStart	0
warmStart	1
linkDown	2
linkUp	3
authenticationFailure	4
egpNeighborLoss	5

You can use the **Gate.SNMP.Specific** property to specify an integer value for the specific trap field in SNMP traps. It can take any integer value between 0 and 2147483647.

The trap message identity is determined based on the values contained in the Enterprises, Standard Trap Type, and Specific Trap Type fields of the Trap Protocol Data Unit (PDU). If the trap type value specified by the **Gate.SNMP.Trap** property is 0, 1, 2, 3, 4, or 5, the trap is a generic trap type and the value of the Specific Trap Type field should be configured to be 0. If the trap type value is 6, the trap is enterprise-specific and is defined in a private Management Information Base (MIB).

Note : The mapfile entry is important if @Class or another field is used. For example:

```
CREATE MAPPING StatusMap
(
   '0' = '@Summary',
   '1' = '@Severity',
   '2' = '@Location',
   '3' = '@Node',
   '4' = '@AlertGroup',
   'Severity' = '@Severity',
<< If Gate.SNMP.Specific:
'@Severity'
   'Class' = '@Class',
<< If Gate.SNMP.Specific:
 '@Class'
   'Node' = '@Node'
)
```

If **Gate.SNMP.Specific** is of the form '@<fieldname>', the the corresponding specific trap value is taken from the map field of '<fieldname>'. For example, if **Gate.SNMP.Specific** is set to the value '@Class' you could define the StatusMap as:

```
CREATE MAPPING StatusMap
(
'0' = '@Summary',
'1' = '@Severity',
'2' = '@Location',
'3' = '@Node',
'4' = '@AlertGroup',
'Class' = '@Class',
'Node' = '@Node'
)
```

A second example, if **Gate.SNMP.Specific** is set to the value '@Severity' you could define the StatusMap as:

```
CREATE MAPPING StatusMap
(
'0' = '@Summary',
'1' = '@Severity',
'2' = '@Location',
'3' = '@Node',
'4' = '@AlertGroup',
'Severity' = '@Severity',
'Node' = '@Node'
)
```

Specifying SNMPv3 support

The gateway supports SNMPv3 traps using the User-Based Security Model (USM).

To use SNMPv3, you must specify values for the following properties:

Table 10. SNMPv3 properties

Property	Description
Gate.SNMP.EngineID	This property specifies the gateway as the source of the SNMPv3 traps.
Gate.SNMP.SecurityName	This property specifies the security name of the gateway as defined in the configuration file of the SNMP receiver.
Gate.SNMP.SecurityLevel	This property specifies the security level that the gateway uses for SNMPv3 messages.
Gate.SNMP.SecurityAuthProtocol	This property specifies the authentication protocol that the gateway uses.
Gate.SNMP.SecurityAuthPassphrase	This property specifies the password required for authentication.
Gate.SNMP.SecurityPrivProtocol	This property specifies the privacy protocol that the gateway uses to encrypt traps.
Gate.SNMP.SecurityPrivPassphrase	This property specifies the password required for privacy.

Controlling the frequency at which the gateway requests updates from the ObjectServer

You can control the rate at which the gateway reader requests updates from the ObjectServer.

You can use the **Gate.Reader.IducFlushRate** property to specify the frequency (in seconds) at which the gateway reader requests updates from the ObjectServer using an Insert, Delete, Update, or Control (IDUC) communication protocol.

The default value of the ObjectServer IDUC update interval is 60 seconds (as specified by the ObjectServer **Granularity** property). This default value is optimal for most systems. The default value of the **Gate.Reader.IducFlushRate** property is 0, which equates to the default ObjectServer update interval. This means that, using the default setting of 0 seconds for the IDUC flush rate and the default setting of 60 for the ObjectServer granularity, the gateway reader gets its updates at 60-second intervals.

You can specify a value greater than 0 for the **Gate.Reader.IducFlushRate**. When the **Gate.Reader.IducFlushRate** is configured to be less than 60 it makes the gateway reader run at a faster rate (higher granularity) than the ObjectServer, thus enabling the gateway to capture more detailed event changes in systems where the ObjectServer itself has high granularity settings.

Note : When both the ObjectServer **Granularity** property and the **Gate.Reader.IducFlushRate** property are set to their default values, the gateway's ability to forward alert updates is affected. If the **Gate.SNMP.ForwardUpdates** property is set to TRUE in these conditions, the gateway will not be able to capture all alert updates.

Forwarding alert updates

The gateway can forward alert data that has been previously sent to the SNMP agent (gateway target).

The default gateway setting is to forward only new alerts to the SNMP agent (gateway target). To forward alert updates, you must specify a value of TRUE for the **Gate.SNMP.ForwardUpdates** property. In effect, this reduplicates alerts (as opposed to the ObjectServer deduplication mechanism).

Note : The gateway never forwards alert deletion information. Journals and details are not processed by the SNMP Gateway.

Controlling the size of the hash table cache

The gateway uses a hash table cache to store details of tables that must be transferred from one ObjectServer to another. The cache aids performance optimization by providing the gateway with an in-memory summarized view of the contents of the ObjectServer to which it is linked.

Using a hash table cache means that the gateway does not have to query an ObjectServer to check for the existence of an event, or the **Serial** value or **Tally** value of an event. Instead, it can check the cache of the target ObjectServer.

The main function of the cache is to facilitate insert operations in Journal and Details tables. When a journal or detail is forwarded for insertion into a target ObjectServer, the gateway writer needs to know the corresponding **Serial** value in the target ObjectServer. This information is found in the cache. It is also used for any other tables specified using the table replication definition table.

You can control the size of the hash table cache using the **Gate.CacheHashTblSize** property. By default, the size of the hash table cache is 5023 elements (rows). This can be increased if the status table has a large number of rows (for example, in excess of 20,000).

Note : To maximize efficiency, you should specify a prime number for the **Gate.CacheHashTblSize** property.

Specifying the gateway to run under Process Agent control

The gateway can be run under Process Agent (PA) control.

The **Gate.PAAware** property specifies whether the gateway is PA-aware. The **Gate.PAAwareName** property specifies which PA is running the gateway.

Note : These properties are maintained automatically by the PA server and provide information only. Do not manually change the values of these properties.

Controlling forwarded TCP/IP Source IP Address

The SNMP Gateway does not appear to support an IPv6 **Gate.SNMP.LocalName**.

Where using UDP protocol only, the gateway provides the property **Gate.SNMP.LocalName** to allow the source IP address of a trap datagram to be sent by the gateway to be configurable, provided it corresponds to an actual network interface for the gateway host.

The property **Gate.SNMP.LocalName** is of type string, the default value is an empty string. This functionality provides support for IPV4 addresses only. Defining property **Gate.SNMP.LocalName**.

Note : The intended use for this property is to specify a fully qualified hostname or IP address that unambiguously identifies the host. Therefore the following condition must be met when defining this property:

- The resolved IP address must correspond to an active interface on the gateway server.

Controlling SNMP Agent Address by Node entry

The Node entry in the snmp.map configuration file is a key configuration item.

This entry determines the SNMPv1 agent-addr value (snmpTrapAddress.0 varbind value in SNMP v2 and v3). The gateway converts this value (either an IP address or a host name) to an IPv4 address, and sets the agent-addr (or snmpTrapAddress.0) value in the SNMP PDU with the resulting address.

Note : The gateway currently only supports IPv4. If the host name resolves to an IPv6 address, the resulting IPv6 address will be ignored and the local IPv4 address used instead.

Startup command file

The startup command file contains a set of commands that the gateway executes each time it starts.

You can specify the location of the startup command file using the generic Netcool/OMNIbus **Gate.StartupCmdFile** property.

The default startup command file, `snmp.startup.cmd`, is located in the following directory: `$OMNIHOME/gates/snmp/`.

The default startup command file contains example commands. You should make a copy of the default file for future reference.

You can use the following commands within the startup command file:

- `GET CONFIG` - Use this command to display the current configuration of the gateway by listing all properties and their values.
- `SET LOG LEVEL TO` - Use this command to set the level of message logging for the gateway. This command can take the following values: `fatal, error, warn, info` or `debug`. The default logging level is `warn`.
- `TRANSFER FROM` - Use this command to initiate a data transfer operation between tables in two ObjectServers.

These commands can also be entered using the SQL interactive interface (`nco_sql`). For more information about using the SQL interactive interface, see the *IBM Tivoli Netcool/OMNIbus Administration Guide*.

For more information about the startup command file, see the *IBM Tivoli Netcool/OMNIbus Probe and Gateway Guide*.

Properties file

The properties file is a text file that contains a set of properties and their corresponding values. These properties define the operational environment of the gateway, such as connection details and the location of the other configuration files.

Properties and command line options

The `NCO_GATE.props` properties file delivered with the SNMP Writer Gateway is a text file that contains a set of properties and their corresponding values. These properties define the operational environment of the gateway, such as connection details and the location of the other configuration files.

The `NCO_GATE.props` properties file contains the categories of properties described in the following sections:

- "Common Tivoli Netcool/OMNIbus properties" on page 18
- "Common gateway properties" on page 19
- "Gateway-specific properties" on page 23

For more information about generic and Inter-Process Communication (IPC) properties and command line options, see the *IBM Tivoli Netcool/OMNIbus Probe and Gateway Guide*.

Common Tivoli Netcool/OMNIbus properties

Table 11 on page 18 describes the available Tivoli Netcool/OMNIbus common properties.

Table 11. Common Netcool/OMNIbus properties

Property name	Command line option	Description
Help *boolean*	`-help` *boolean*	Use this property to instruct the gateway to display application help information on startup and exit. The default is FALSE.
MaxLogFileSize *integer*	`-maxlogfilesize` *integer*	Use this property to specify the size (in bytes) that the gateway allocates for the log file. When the log file reaches this size, the gateway renames the log file by appending the name with the suffix `.old` and creates a new log file. The default is `1024`.
MessageLevel *string*	`-messagelevel` *string*	Use this property to specify the reporting level of the log file messages. The default is `warn`.

Table 11. Common Netcool/OMNIbus properties (continued)		
Property name	**Command line option**	**Description**
MessageLog *string*	`-messagelog` *string*	Use this property to specify the location and name of the message log file. The default is `$OMNIHOME/log/NCO_GATE.log`.
Name *string*	`-name` *string*	Use this property to specify the name of the current gateway instance. If you want to run multiple gateways on one machine, you must use a different name for each instance. The default is `NCO_GATE`.
PropsFile *string*	`-propsfile` *string*	Use this property to specify the location and name of the gateway properties file. The default is `$OMNIHOME/etc/NCO_GATE.props`.
UniqueLog *boolean*	`-uniquelog` *boolean*	Use this property to specify whether log file names are made unique by adding the Process ID (PID) of the gateway to the file name. The default is `FALSE`.
Version *boolean*	`-version` *boolean*	Use this property to instruct the gateway to display information about the application version on startup and exit. The default is `FALSE`.

Common gateway properties

Table 12 on page 19 describes the available common gateway properties.

Table 12. Common gateway properties		
Property name	**Command line option**	**Description**
ConfigCryptoAlg *string*	`-configcryptoalg` *string*	Use this property to specify the cryptographic algorithm to use when encrypting and decrypting config values and files. The default is `AES`.

Table 12. Common gateway properties (continued)

Property name	Command line option	Description
ConfigKeyFile *string*	-configkeyfile *string*	Use this property to specify the location of the file containing the key for encrypted config values and files.
Gate.CacheHashTblSize *integer*	-chashtblsize *integer*	Use this property to specify the number of elements (database rows) that the gateway allocates for the hash table cache. The default is 5023.
Gate.MapFile *string*	-gatemap *string*	Use this property to specify the location and name of the mapping file. The default is $OMNIHOME/gates/snmp/snmp.map.
Gate.Mapper.Debug *boolean*	-mapperdebug *boolean*	Use this property to specify whether or not the gateway includes mapper debug messages in the debug log. The default is TRUE.
Gate.Mapper.ForwardHistoricDetails *boolean*	-mapperforhistdtls *boolean*	The default is FALSE. This feature is no longer supported.
Gate.Mapper.ForwardHistoricJournals *boolean*	-mapperforhistjrnl *boolean*	The default is FALSE. This feature is no longer supported.
Gate.NGtkDebug *boolean*	-ngtkdebug *boolean*	Use this property to specify whether or not the NGTK library logs debug messages. The default is TRUE.
Gate.PAAware *integer*	-paaware *integer*	This property indicates whether or not the gateway is Process Agent (PA) aware. The default is 0 (the gateway is not PA-aware). **Note :** This property is maintained by the PA server and is included in the properties file for information only.

Table 12. Common gateway properties (continued)		
Property name	**Command line option**	**Description**
Gate.PAAwareName *string*	-paname *string*	This property indicates the name of the PA controlling the gateway. The default is "". **Note :** This property is maintained by the PA server and is included in the properties file for information only.
Gate.Reader.Debug *boolean*	-readerdebug *boolean*	Use this property to specify whether the gateway includes gateway reader debug messages in the debug log. The default is TRUE.
Gate.Reader.Description *string*	-readerdescription *string*	Use this property to specify the application description for the reader connection. This description is used in triggers and allows you to determine which component of the gateway attempted to perform an action. The default is Gateway Reader.
Gate.Reader. DetailsTableName *string*	-readerdetailstblname *string*	The default is alerts.details. This feature is no longer supported.
Gate.Reader. JournalTableName *string*	-readerjournaltblname *string*	The default is alerts.journal. This feature is no longer supported.
Gate.Reader. FailbackEnabled *boolean*	-failbackenabled *boolean*	Use this property to specify whether or not the gateway attempts to fail back to the primary system following a system failover. The default is TRUE. **Note :** The gateway attempts to fail back with the frequency specified by the **Gate.Reader. FailbackTimeout** property.

Table 12. Common gateway properties (continued)		
Property name	**Command line option**	**Description**
Gate.Reader. FailbackTimeout *integer*	`-failbacktimeout` *integer*	Use this property to specify the frequency (in seconds) with which the gateway attempts to fail back to the primary system following a system failover. The default is `30`. **Note :** The gateway attempts to fail back to the primary ObjectServer only if the **Gate.Reader. FailbackEnabled** property is set to TRUE.
Gate.Reader.IducFlushRate *integer*	`-readeriducflushrate` *integer*	Use this property to specify the frequency (in seconds) at which the gateway reader requests updates from the ObjectServer. The default is `0`. **Note :** If you set this property to a value greater than 0, the reader issues automatic IDUC flush requests to the ObjectServer with greater frequency. See "Controlling the frequency at which the gateway requests updates from the ObjectServer" on page 16 for more information.
Gate.Reader.LogOSSql *boolean*	`-readerlogossql` *boolean*	Use this property to specify whether the gateway logs all SQL commands sent to the ObjectServer in debug mode. The default is FALSE.
Gate.Reader.Password *string*	`-readerpassword` *string*	Use this property to specify the password associated with the user specified by the **Gate.Reader.Username** property. The default is `""`. **Note :** This password must be encrypted using the `nco_g_crypt` utility. For information about using `nco_g_crypt`, see the *IBM Tivoli Netcool/OMNIbus Administration Guide*, (SC23-6371).

Table 12. Common gateway properties (continued)		
Property name	**Command line option**	**Description**
Gate.Reader. ReconnectTimeout *integer*	`-readerreconntimeout` *integer*	Use this property to specify the time (in seconds) between each reconnection poll attempt that the gateway makes if the connection to the ObjectServer is lost. The default is `30`.
Gate.Reader.Server *string*	`-readerserver` *string*	Use this property to specify the name of the ObjectServer from which the gateway reads alerts. The default is `NCOMS`.
Gate.Reader. StatusTableName *string*	`-readerstatustblname` *string*	Use this property to specify the name of the status table that the gateway reads. The default is `alerts.status`.
Gate.Reader.Tbl ReplicateDefFile *string*	`-readertblrepdef` *string*	Use this property to specify the location of the table replication definition file. The default is `$OMNIHOME/gates/snmp/snmp.reader.tblrep.def`.
Gate.Reader.Username *string*	`-readerusername` *string*	Use this property to specify the user name that is used to authenticate the ObjectServer connection. The default is `root`.
Gate.StartupCmdFile *string*	`-startupcmdfile` *string*	Use this property to specify the location of the startup command file. The default is `$OMNIHOME/gates/snmp/snmp.startup.cmd`.
Gate.Transfer. FailoverSyncRate *integer*	`-fsyncrate` *integer*	Use this property to specify the rate (in seconds) of the failover synchronization. The default is `60`.

Gateway-specific properties

Table 13 on page 24 describes the available properties specific to the SNMP Writer Gateway.

Table 13. Gateway-specific properties		
Property name	**Command line option**	**Description**
Gate.SNMP.Community *string*	`-snmpcommunity` *string*	Use this property to specify the community string for SNMP traps. The default is `public`.
Gate.SNMP.EnableLookup *boolean*	`-snmpenablelookup` *boolean*	Use this property to specify the whether or not the lookup of the Node value is enabled. The default is TRUE.
Gate.SNMP.EngineID *string*	`-snmpengineid` *string*	Use this property to specify the engine ID of the gateway. This identifies the gateway as the source of the SNMPv3 traps. The default is `0x0102030405`. **Note :** This property is only used with SNMPv3 traps and must match the engine ID specified in the configuration file of the receiver.
Gate.SNMP.ForwardUpdates *boolean*	`-snmpforwardupdates` *boolean*	Use this property to specify whether or not the gateway forwards alert updates to SNMP agent (gateway target). In effect, the original alert is duplicated but will include the updated data. The default is FALSE.
Gate.SNMP.Gateway *string*	`-snmpgateway` *string*	Use this property to specify the IP address and port to which the gateway forwards traps. The default is `127.0.0.1:162`. **Note :** If you are operating in an IPv4 environment, specify this location in IPv4 format as *address*:*port*. For example: `127.0.0.1:8080` If you are operating in an IPv6 environment, specify this location in IPv6 format, preceded by tcp6 or udp6 (as appropriate), and followed by the port number, as *tcp6*|*udp6*:*address*:*port*. For example: `tcp6:[::01]:6666`

Table 13. Gateway-specific properties (continued)		
Property name	**Command line option**	**Description**
Gate.SNMP.LocalName *string*	-snmplocalname *string*	Use this property to specify the sender full qualified domain name or IP address of the override. The default is "". For valid values refer to the "Controlling forwarded TCP/IP Source IP Address" on page 17.
Gate.SNMP.OID *string*	-snmpoid *string*	Use this property to specify the object identifier (OID) for traps. The default is 1.3.6.1.4.1.1279 (this is an IANA-registered Private Enterprise Number). **Note :** This property can also be defined as @NodeGroup to forward the value of the NodeGroup column in the status table.
Gate.SNMP.Protocol *string*	-snmpprotocol *string*	Use this property to specify the transport protocol that the gateway uses. This property takes the following values: TCP: The gateway uses Transmission Control Protocol. UDP: The gateway uses User Datagram Protocol. The default is UDP. **Note :** Store-and-forward mode is not available when the gateway is using UDP. See "Store-and-forward mode" on page 9 for more information.

Table 13. Gateway-specific properties (continued)

Property name	Command line option	Description
Gate.SNMP.Retries *integer*	`-snmpretries` *integer*	Use this property to specify the number of times that the gateway attempts to retry sending a message on failure. When this number is exceeded, the gateway stops sending the messages to the port. For example: The gateway retries **Gate.SNMP.Retries** = x + 1 times. Therefore if **Gate.SNMP.Retries** = 5, the gateway retries 6 times. The default is 5. **Note :** Retries only apply to situation where **Gate.SNMP.Protocol** property is set to 'TCP'.
Gate.SNMP.RetryInterval *integer*	**-snmpretryinterval** *integer*	Use this property to specify how long (in seconds) the gateway waits before retrying sending an SNMP trap following a previous failure. The default is 3.
Gate.SNMP.SecurityLevel *string*	`-snmpsecuritylevel` *string*	Use this property to specify the security level that the gateway uses for SNMPv3 messages. This property takes the following values: `AuthnoPriv`: The gateway sends the username and password in encrypted format. `AuthPriv`: The gateway transmits the SNMP traps in encrypted format. `noAuthnoPriv`: The gateway does not encrypt the username, the password, or the SNMP traps. The default is `noAuthnoPriv`. **Note :** This property is only used with SNMPv3 traps.

Table 13. Gateway-specific properties (continued)		
Property name	**Command line option**	**Description**
Gate.SNMP.SecurityName *string*	`-snmpsecurityname` *string*	Use this property to specify the security name for the gateway as defined in the configuration file of the receiver. The default is `netcool`. **Note :** This property is only used with SNMPv3 traps.
Gate.SNMP.SecurityAuthProtocol *string*	`-snmpsecurityauthprotocol` *string*	Use this property to specify the authentication protocol that the gateway uses. This property takes the following values: MD5: The gateway uses the Message Digest 5 protocol. SHA, SHA1, and SHA256: The gateway uses the Secure Hash Algorithm protocol. The default property value is MD5. If an invalid property value is configured, the gateway will use the SNMP default protocol of SHA1. **Note :** This property is only used with SNMPv3 traps. MD5 is not permitted when FIPS mode is enabled in the OMNIbus environment. SHA and SHA1 are equivalent.
Gate.SNMP.SecurityPrivProtocol *string*	`-snmpsecurityprivprotocol` *string*	Use this property to specify the privacy protocol that the gateway uses to encrypt data. This property takes the following values: AES, AES192, and AES256: The gateway uses the Advanced Encryption Standard. DES: The gateway uses the Data Encryption Standard. The default property value is DES. If an invalid property value is configured, the gateway will use the SNMP default protocol of AES. **Note :** DES is not permitted when FIPS mode is enabled in the OMNIbus environment.

Table 13. Gateway-specific properties (continued)

Property name	Command line option	Description
Gate.SNMP.SecurityAuthPassphrase *string*	-snmpsecurityauthpassphrase *string*	Use this property to specify the password used for authentication. The default is password. **Note :** The password must be at least eight characters long. This property is only used with SNMPv3 traps.
Gate.SNMP.SecurityPrivPassphrase *string*	-snmpsecurityprivpassphrase *string*	Use this property to specify the password used for privacy. The default is password. **Note :** This property is only used with SNMPv3 traps.
Gate.SNMP.SNMPVersion *integer*	-snmpsnmpversion *integer*	Use this property to specify the version of the SNMP writer. **Note :** Only three options are supported: 1 [SNMPv1] 2 [SNMPv2] 3 [SNMPv3] The default is 2.
Gate.SNMP.Specific *string*	-snmpspecific *string*	Use this property to specify the trap type value for the specific trap field in forwarded SNMP traps. The default is 1. See "Specifying values for the generic and specific fields in SNMP traps" on page 14 for more information. **Note :** This property can also be defined as @<column name> to forward the value of the column name column in the alerts.status table.
Gate.SNMP.Store AndForward *boolean*	-snmpstoreandforward *boolean*	The default is FALSE. This feature is no longer supported.
Gate.SNMP.StoreFile *string*	-snmpstorefile *string*	The default is $OMNIHOME/var/NCO_GATE_snmp_.store. This feature is no longer supported.

Table 13. Gateway-specific properties (continued)		
Property name	**Command line option**	**Description**
Gate.SNMP.Timeout *integer*	-snmptimeout *integer*	Use this property to specify the time (in seconds) that the gateway waits for a connection from an SNMP receiver before timing out. The default is 600. **Note :** This property is only used when the **Gate.SNMP.Protocol** property is set to TCP.
Gate.SNMP.Trap *string*	-snmptrap *string*	Use this property to specify the trap type value of the generic trap field in forwarded SNMP traps. The default is "6". The value is restricted from 0 to 6. See "Specifying values for the generic and specific fields in SNMP traps" on page 14 for more information. **Note :** This property can also be defined as @Severity to forward the value of the Severity column in the alerts.status table.

Note :

Despite their presence in -help and -dumprops output, the following properties are not used in the gateway:

- -snmpfailbackenabled
- -snmpfailbacktimeout
- -snmpreconntimeout
- Gate.SNMP.FailbackEnabled
- Gate.SNMP.FailbackTimeout
- Gate.SNMP.ReconnectTimeout

Running the gateway

This topic describes how to run the gateway on UNIX and Windows operating systems. On Windows operating systems, you can run Process Control as a service and configure it to run the gateway.

Before running the gateway on UNIX and Windows operating systems, do the following:

1. Copy the example properties file, NCO_GATE.props, from the $OMNIHOME/gates/snmp directory to the following directory: $OMNIHOME/etc.
2. Make a backup copy of the NCO_GATE.props prior to making any edits.
3. Edit the properties in the NCO_GATE.props file to suit your environment. See "Configuring the gateway" on page 5 and "Gateway operation" on page 12 for guidance on editing properties to suit your environment.

To start the gateway on UNIX and Linux operating systems, run the following command:

```
$OMNIHOME/bin/nco_g_snmp
```

To start the gateway as a process on Windows operating systems, run the following command:

```
%OMNIHOME%\bin\win32\nco_g_snmp.exe
```

More updates may be required for different Windows environments.

To start the gateway as a service on Windows operating systems, use the following steps:

1. Register the gateway with the Service Control Manager.
2. If the gateway and the ObjectServer are running on the same host, run the following command:

   ```
   %OMNIHOME%\bin\nco_g_snmp.exe -install -depend NCOObjectServer
   ```
3. If the gateway and the ObjectServer are running on different hosts, run the following command:

   ```
   %OMNIHOME%\bin\nco_g_snmp.exe -install
   ```
4. Start the gateway using the Microsoft Services Management Console.

Error handling

You can troubleshoot problems with the gateway by consulting error messages. To help you do this, the gateway has configurable error handling.

Error handling is provided by the Tivoli Netcool/OMNIbus Gateway Toolkit (NGTK) library. To specify that the NGTK library logs debug messages, set the **Gate.NGtkDebug** property to TRUE.

Error messages

Error messages provide information about problems that have occurred during the operation of the gateway. You can use the information that they contain to resolve such problems.

Cannot connect to vmwtpm0940.hursley.ibm.com:1621 (Connection refused).

The following table describes the error messages that the gateway generates:

Table 14. Error messages

Error	Description	Action
`Failed to send trap for SNMP trap forwarder with error error. Retrying...`	The gateway failed to forward a trap for the reason specified in the error message. It is trying to re-send the trap.	Verify that the endpoint is available and reachable. Check for any intervening firewalls.
`Failed to create session for SNMP trap forwarder.`	The gateway was unable to reconnect to the endpoint after a previous disconnection.	Verify that the endpoint is still available and that there is no intervening firewall.
`Failed to initialize gateway properties.`	The gateway could not initialize the gateway properties.	Verify that the properties file has correct permissions. Check the values set in the properties file.
`Invalid value specified for PROTOCOL attribute. Ignored and using UDP.`	The transport protocol field is incorrectly specified and the gateway is defaulting to the UDP protocol.	Specify the value of the **Gate.SNMP.Protocol** property as either UDP or TCP.
`Node index not found for SNMP trap forwarder`	The gateway was unable to find a column entitled 'Node' in the configured table.	Verify that there is a `'Node' = '@<column_name>'` entry in the mapping file that you are using.

Table 14. Error messages (continued)

Error	Description	Action
`oid property has an invalid value; must be a string or a field reference.`	The **`Gate.SNMP.OID`** property must be specified as an object identifier (OID) string or a reference to the node group field.	Ensure that the **`Gate.SNMP.OID`** property is specified as an OID string or as `@NodeGroup`.
`Snmpversion property has unknown SNMP version specified in SNMP trap forwarder. Please set it to 1 2 or 3`	The SNMP writer version is incorrectly specified.	Specify the value of the **`Gate.SNMP.SNMPVersion`** property as either 1, 2, or 3.
`Specific property has an invalid value; must be a number or a field reference.`	The value for the specific trap field in forwarded SNMP traps is incorrectly specified.	Specify the **`Gate.SNMP.Specific`** property as a string representing an integer value. Either directly e.g. '1' or via a field reference e.g. `'@Class'`.
`Security level property has an invalid value; must be one of the following: noAuthnoPriv AuthnoPriv AuthPriv`	The security level that the gateway uses for SNMPv3 messages is incorrectly specified.	Specify one of the following values for the **`Gate.SNMP.SecurityLevel`** property: • `noAuthnoPriv` • `AuthnoPriv` • `AuthPriv`
`Setting security level to noAuthnoPriv`	The gateway is setting the security level for SNMPv3 messages to `noAuthnoPriv`. At this security level, the gateway does not encrypt the username, the password, or the SNMP traps.	The gateway is setting the security level for SNMPv3 messages to `noAuthnoPriv`. You can set the security level to `AuthnoPriv` or `AuthPriv` using the **`Gate.SNMP.SecurityLevel`** property.
`Security auth protocol property has an invalid value; must be one of the following: SHA1 MD5`	The authentication protocol that the gateway uses for SNMPv3 traps is incorrectly specified.	Specify one of the following values for the **`Gate.SNMP.Security AuthProtocol`** property: • `MD5` • `SHA` • `SHA1` • `SHA256`

Table 14. Error messages (continued)

Error	Description	Action
`Security priv protocol property has an invalid value; must be one of the following: DES AES. Setting security priv protocol to DES`	The privacy protocol (encryption standard) that the gateway uses to encrypt data is incorrectly specified. The gateway is setting the protocol to DES.	The gateway is setting the privacy protocol to DES. You can make the gateway use AES encryption by setting the value of the **`Gate.SNMP.Security PrivProtocol`** property to one of the following values: • AES • AES192 • AES256
`Security auth phrase must be at least 8 characters long`	The password that the gateway uses for SNMPv3 authentication is not long enough.	Specify the value of the **`Gate.SNMP.Security AuthPassphrase`** property as a string at least eight characters long.
`SNMP Writer '`*`writer`*`': Failed to write record to SAF store file.`	The SNMP writer failed to write a SAF record to the file specified.	Verify that the file is writable and that the device is not full.
`Trap property has an invalid value; must be integer or field reference.`	The value for the generic trap field in SNMP forwarded traps is incorrectly specified.	Specify the value of the **`Gate.SNMP.Trap`** property as either an integer or as `@Severity`.
`Failed to create session for SNMP trap forwarder.`	If **`Gate.SNMP.LocalName`** is not assigned a valid IP Address on the SNMP gateway Server, the following error is reported in the gateway logfile and the gateway fails to start.	Check **`Gate.SNMP.LocalName`** property value (9.20.66.25) is valid.

Known issues

At the time of release, some issues were reported that you should be aware of when running the gateway. This section contains information about these known issues.

The first SNMP message sent to a receiver does not result in SNMP retries

The first SNMP message sent to a receiver which is shut down does not result in SNMP retries. The gateway relies on the API to inform it of failure. The API does not register the first failure as a fail.

Gateway retries connecting to unknown host

The gateway tries to connect to an unknown hostname multiple times during startup. Users receive the following error message:

```
@: Unknown host (vmjim:1621) (Connection refused)
```

Note : Due to restrictions in the SNMP v1 trap agent address field, only IPv4 addresses are supported as agent addresses.

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.

Part Number:

(1P) P/N:

SC23-7804-10