

IBM® Tivoli® Netcool/OMNIbus Gateway for
Tivoli EIF
5.0

Reference Guide
August 6, 2015



Notice

Before using this information and the product it supports, read the information in [Appendix A, “Notices and Trademarks,”](#) on page 35.

Edition notice

This edition (SC23-9117-07) applies to version 5.0 of IBM Tivoli Netcool/OMNIbus Gateway for Tivoli EIF and to all subsequent releases and modifications until otherwise indicated in new editions.

This edition replaces SC23-9117-06.

© **Copyright International Business Machines Corporation 2009, 2015.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

About this guide.....	v
Document control page.....	v
Conventions used in this guide.....	vi
 Chapter 1. Gateway for Tivoli EIF.....	1
Summary.....	1
Overview.....	2
About the Gateway.....	3
Installing the gateway.....	3
Installing probes and gateways on Tivoli Netcool/OMNIBus V8.1.....	3
Installing the gateway on Tivoli Netcool/OMNIBus V7.4.0.....	5
Configuring the gateway.....	6
Specifying a gateway authentication method.....	9
Connecting the gateway to the ObjectServer.....	9
Loading the ObjectServer customization file.....	9
Configuring the gateway to run with Tivoli EIF.....	10
Replicating data using the table replication definition file.....	11
Mapping data using the map definition file.....	12
Store and forward.....	14
Startup command file.....	14
FIPS mode and encryption.....	14
Gateway operations.....	15
Understanding gateway startup operations.....	15
Understanding gateway resynchronization operations.....	15
Identifying events sent by the gateway.....	15
Understanding gateway error handling.....	25
Understanding how the gateway runs under process agent control.....	25
Understanding the Tivoli EIF buffer cache.....	25
Understanding how the gateway controls buffer size.....	25
Understanding how the gateway handles fallback.....	26
Running the Gateway.....	26
Running the Gateway on UNIX.....	26
Running the Gateway on Windows.....	26
Properties and command line options.....	27
Known issues.....	34
 Appendix A. Notices and Trademarks.....	35
Notices.....	35
Trademarks.....	36

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 Gateway for Tivoli EIF 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
SC23-6096-00	December 19, 2006	First IBM publication.
SC23-6096-01	December 22, 2007	Summary section updated. Gate.TIVOLI{EIF.UpdateClass property added.
SC23-9117-01	August 22, 2008	Support for Linux® for zSeries added.
SC23-9117-02	October 31, 2009	Guide updated to describe running the probe in UTF-8 mode on Windows. “Summary” on page 1 updated. “Properties and command line options” on page 27 updated. Description for the -utf8enabled command line option added.
SC23-9117-03	September 30, 2011	Summary section updated. “Configuring the gateway to run with Tivoli EIF” on page 10 updated.
SC23-9117-04	November 30, 2012	Guide updated for Netcool/OMNIBus V7.4 release. “Installing the gateway” on page 3 updated.
SC23-9117-05	November 8, 2013	“Summary” on page 1 updated. “FIPS mode and encryption” on page 14 added to the guide. “Startup command file” on page 14 updated. “Replicating data using the table replication definition file” on page 11 added to the guide.

Table 1. Document modification history (continued)		
Document version	Publication date	Comments
SC23-9117-06	April 02, 2015	“Summary” on page 1 updated. “Connecting the gateway to the ObjectServer” on page 9 updated. “Properties and command line options” on page 27 updated. “Store and forward” on page 14 updated. “Configuring the gateway to run with Tivoli EIF” on page 10 updated. Added an Optimizing the eif_gw_resync trigger section to the “Replicating data using the table replication definition file” on page 11 topic.
SC23-9117-07	August 6, 2015	Updates related to new properties, gateway configuration, and gateway operation. “Properties and command line options” on page 27 updated with these new properties: Gate.TIVOLI{EIF.MaxEventSize, Gate.TIVOLI{EIF.LastState, Gate.TIVOLI{EIF.changeFile, Gate.TIVOLI{EIF.MaxEventSize, Gate.TIVOLI{EIF.MaxIDListSize, and Gate.TIVOLI{EIF.MinimumSeverity.

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 <i>arch</i>
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. Gateway for Tivoli EIF

The Gateway for Tivoli EIF is used to forward events from Tivoli Netcool/OMNIBus to applications that accept events in EIF format.

You use the gateway when the applications receive events from Tivoli Netcool/OMNIBus and other event sources through the Tivoli Event Integration Facility (Tivoli EIF).

This guide contains the following sections:

- [“Summary” on page 1](#)
- [“Overview” on page 2](#)
- [“About the Gateway” on page 3](#)
- [“Installing the gateway” on page 3](#)
- [“Configuring the gateway” on page 6](#)
- [“Gateway operations” on page 15](#)
- [“Running the Gateway on UNIX” on page 26](#)
- [“Running the Gateway on Windows” on page 26](#)
- [“Properties and command line options” on page 27](#)
- [“Known issues” on page 34](#)
- [Appendix A, “Notices and Trademarks,” on page 35](#)

Summary

Use this summary information to learn about this gateway

The following table provides a summary of the Gateway for Tivoli EIF.

Table 3. Summary	
Gateway target	Applications that can read events in EIF format.
Gateway executable name	nco_g_tivoli_eif
Package Version	5.0
Gateway supported on	For details of supported operating systems, see the following Release Notice on the IBM Software Support website: https://www-304.ibm.com/support/docview.wss?uid=swg21613134 Note : This Gateway is not currently supported for HP-UX version 11.11 running on dual processor hardware.

Table 3. Summary (continued)	
Configuration files	Map definition file: \$OMNIHOME/gates/tivoli_eif/tivoli_eif.map Properties file: \$OMNIHOME/gates/tivoli_eif/NC0_GATE.props Note : It is likely that you will rename the NC0_GATE.props properties file. However, this guide will use NC0_GATE.props when referring to the properties file. Table replication definition file: \$OMNIHOME/gates/tivoli_eif/tivoli_eif.rdrwtr.tblrep.def Startup command file: \$OMNIHOME/gates/tivoli_eif/tivoli_eif.startup.cmd ObjectServer customization file: \$OMNIHOME/gates/tivoli_eif/tivoli_eifgw_setup.sql File to configure the Tivoli Event Integration Facility (Tivoli EIF): \$OMNIHOME/gates/tivoli_eif/tivoli_eif_config For information about Tivoli EIF, see the <i>IBM Tivoli Netcool/OMNIBus Event Integration Facility Reference</i>.
Requirements	A currently supported version of IBM Tivoli Netcool/OMNIBus.
Remote connectivity	Available
SSL support	Not available
Multicultural support	Not 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 <i>IBM Tivoli Netcool/OMNIBus Installation and Deployment Guide</i>.

Overview

The Gateway for Tivoli EIF provides an interface between the Tivoli Event Integration Facility (Tivoli EIF) and Tivoli Netcool/OMNIBus.

The following topic promotes an understanding of the gateway and how it interacts with Tivoli EIF:

- [“About the Gateway” on page 3](#)

About the Gateway

The IBM Tivoli Netcool/OMNIBus Gateway for Tivoli EIF uses the IBM Tivoli Netcool/OMNIBus Gateways Toolkit (NGTK) library, which provides the basic framework for the gateway process. The NGTK library is installed as a part of Tivoli Netcool/OMNIBus.

Gateway type

The Gateway for Tivoli EIF is a unidirectional gateway that enables events to flow in one direction, from a source ObjectServer to a Tivoli Event Integration Facility (Tivoli EIF) event consumer. Changes made in the source ObjectServer are reflected in Tivoli EIF, but changes in Tivoli EIF are not reflected in the source ObjectServer.

See the *IBM Tivoli Netcool/OMNIBus Probe and Gateway Guide* for more information about unidirectional gateways.

Information flow

The gateway is comprised of reader/writer, mapper, and writer components. The flow of information between the components is as follows:

1. The reader/writer component reads from the source ObjectServer and passes the data to the mapper component.
2. The mapper component receives data from the reader component, transforms the data into an appropriate form for the target writer using the map definition file (`tivoli_eif.map`), and passes the data to the writer component.
3. The writer component receives the source data from the mapper component and writes it to Tivoli EIF.

Note : The gateway adds a new Tivoli EIF attribute (`gateway_instance`) to each Tivoli Netcool/OMNIBus event sent to Tivoli EIF. This enables you to identify which gateway sent which events to Tivoli EIF. The values of the `gateway_instance` attribute is determined by the **Gate.TIVOLI_EIF.GWInstanceName** property defined in the `NCO_GATE.props` properties file.

Centralized property management

The gateway uses centralized property management and separates properties from data processing configuration. The gateway can be configured using configuration files; for details, see [“Configuring the gateway”](#) on page 6.

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 -  
installationDirectory /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 -  
installationDirectory /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**.

Installing the gateway on Tivoli Netcool/OMNIBus V7.4.0

For Tivoli Netcool/OMNIBus V7.4.0, all gateways are installed using the Tivoli Netcool/OMNIBus installer.

You can install the gateway using any of the following:

- “The installation wizard” on page 5
- A text-based installer: “Console mode” on page 6
- Settings that are predefined in a text file: “Silent mode” on page 6

The installation package and patches for the gateway are supplied as archives. The archive management application that you use to extract the files must be able to preserve the directory structure contained in the archive on extraction.

Note : If you are installing a 32-bit gateway on a system that runs a 64-bit UNIX or Linux operating system, you will need to install additional, 32-bit operating system libraries. See the *IBM Tivoli Netcool/OMNIBus Installation and Deployment Guide* for more information.

Obtaining the installation package

To obtain the installation package and prepare it for installation, use the following steps:

1. Download the installation package for the gateway from the Passport Advantage Online Web site:
http://www-306.ibm.com/software/howtobuy/passportadvantage/pao_customers.htm
2. Make a backup of any existing configuration files that you want to retain.
3. Extract the contents of the installation package to a temporary directory.

Now use one of the installation methods to install your gateway. In each case, the gateway is installed in the following directory:

On Linux and UNIX: `$NCHOME/omnibus/gates`

On Windows: `%NCHOME%\omnibus\gates`

The installation wizard

To install the gateway using the installation wizard:

1. Run the installer for your operating system:

- On Linux and UNIX: `$NCHOME/omnibus/install/nco_install_integration`
On Windows: `%NCHOME%\omnibus\install\nco_install_integration`
2. When the installation wizard starts, specify the extracted directory that contains the `README.txt` file as the location of the gateway installation files.
 3. Accept the license conditions.

Console mode

To install the gateway in console mode:

1. Run the installer for your operating system:
On Linux and UNIX: `$NCHOME/omnibus/install/nco_install_integration -i console`
On Windows: `%NCHOME%\omnibus\install\nco_install_integration -i console`
2. When the text-based installer starts, specify the extracted directory that contains the `README.txt` file as the location of the gateway installation files.
3. Accept the license conditions.

Silent mode

To install the gateway in silent mode:

1. Create a text file named `reponse.txt` and add the following entries:

```
PROBE_OR_GATE_LOCATION=README_directorypath  
LICENSE_ACCEPTED=true
```

where `README_directorypath` is the path to the directory containing the `README.txt` file in the extracted package.

2. Run the installer for your operating system:
On Linux and UNIX: `$NCHOME/omnibus/install/nco_install_integration -i silent -f response_path/response.txt`
On Windows: `%NCHOME%\omnibus\install\nco_install_integration -i silent -f response_path\response.txt`

where `response_path` is the full path to the `response.txt` file.

Configuring the gateway

After installing the gateway you need to make various configuration settings to suit your environment. The gateway provides a set of configuration files that you use to specify the settings for your environment.

Understanding the gateway configuration files

Configuration files define the environment in which the gateway operates and how the gateway executes relative to performing such tasks as resynchronization. A gateway typically provides these configuration files: properties file, map definition file, startup command file, and table replication definition file. A gateway can also provide configuration files specific to its operation. For example, the Gateway for Tivoli EIF provides a configuration file called `tivoli_eif_config`.

Specifically, the Gateway for Tivoli EIF provides the configuration files listed in [Table 4 on page 7](#). See the *IBM Tivoli Netcool/OMNIBus Probe and Gateway Guide* for additional information about each of the configuration file types identified in the first column of the table.

Table 4. Description of Gateway for Tivoli EIF configuration files		
Configuration file type	Configuration file name for Gateway for Tivoli EIF	Purpose
Properties file		
Map definition		
Startup command file		The startup command file allows for setting up data operations to occur automatically when the gateway starts
Table replication definition file		
File to configure the Tivoli Event Integration Facility (Tivoli EIF)	\$OMNIHOME/gates/tivoli_eif/tivoli_eif_config	The tivoli_eif_config file contains properties used to configure the Tivoli EIF.

Note : All users, groups, roles, and restriction filters must be identical in the primary ObjectServer and secondary ObjectServer.

Understanding gateway configuration tasks

The following table 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.

Table 5. Configuring the gateway		
Configuration tasks	Properties	See
Required configuration tasks:		
Authentication configuration task		
Authenticate the gateway	If specifying PAM authentication, use the Gate.UsePamAuth property in the NCO_GATE.props properties file.	“Specifying a gateway authentication method” on page 9
Connection Configuration Tasks		

Table 5. Configuring the gateway (continued)

Configuration tasks	Properties	See
Connect the gateway to the ObjectServer	Specify the ObjectServer to which the gateway connects by setting the Gate.RDRWTR.Server property in the <code>NCO_GATE.props</code> properties file.	“Connecting the gateway to the ObjectServer” on page 9
Connect the gateway to Tivoli EIF	Use the Gate.TIVOLI_EIF.Config property in the <code>NCO_GATES.props</code> properties file to specify the path and the gateway specific configuration file. The default file is called <code>tivoli_eif_config</code> . In the <code>tivoli_eif_config</code> file, specify these properties: c1ServerLocation , c1Port , and BufEvtPath .	“Configuring the gateway to run with Tivoli EIF” on page 10
Load the ObjectServer customization file.	None	“Loading the ObjectServer customization file” on page 9
Optional configuration tasks:		
Table replication configuration task		
Define the tables and event types that are replicated between the ObjectServer and Tivoli EIF.	Gate.RdrWtr.TblReplicateDefFile	“Replicating data using the table replication definition file” on page 11
Data mapping configuration task		
Define how fields in ObjectServer tables map to fields in Tivoli EIF.	Use the Gate.Mapfile property in the <code>NCO_GATE.props</code> properties file to specify the path and name of the map definition file. Then, adjust the mapping in this map definition file to suit your requirements.	“Mapping data using the map definition file” on page 12
Store and forward configuration task		
Configure the gateway to support store and forward functionality as required. Note : The store and forward functionality is enabled by default.	Use the Gate.TIVOLI_EIF.Config property in the <code>NCO_GATES.props</code> properties file to specify the gateway specific configuration file called <code>tivoli_eif_config</code> . In the <code>tivoli_eif_config</code> file, specify the appropriate properties.	“Store and forward” on page 14

Table 5. Configuring the gateway (continued)		
Configuration tasks	Properties	See
ObjectServer customization file task		
Load the ObjectServer customization file (tivoli_eifgw_setup.sql).	None	“Loading the ObjectServer customization file” on page 9
FIPS mode and encryption configuration tasks		
Operate the gateway using FIPS mode and encryption.	None	“FIPS mode and encryption” on page 14
Operate the gateway using AES encryption.	None	“FIPS mode and encryption” on page 14

Specifying a gateway authentication method

The Gateway for Tivoli EIF supports standard UNIX authentication or Pluggable Authentication Modules (PAM) authentication.

By default, the gateway uses standard UNIX authentication. To use PAM authentication, set the **Gate.UsePamAuth** property to TRUE.

Connecting the gateway to the ObjectServer

To enable communication between the gateway and the ObjectServer, configure communication details for the ObjectServer and the gateway using the Tivoli Netcool/OMNIBus Server Editor and create an entry for the ObjectServer in the interfaces file (\$NCHOME/etc/omni.dat).

Create the server NCO_GATE in the interfaces file using the Tivoli Netcool/OMNIBus Server Editor.

On UNIX and Linux operating systems, use the following command to start the Server Editor:

```
$NCHOME/omnibus/bin/nco_xigen
```

On Windows operating systems, use the following to start the Server Editor:

Start > Programs > NETCOOL Suite > System Utilities > Servers Editor

Note : If there is a firewall between the gateway and the ObjectServer, configure the ObjectServer to use a fixed port for IDUC and ensure that both the main ObjectServer port and the IDUC port are opened in the firewall. By default, the ObjectServer uses a random IDUC port.

For more information about using the Server Editor, the interfaces file, and the nco_xigen utility see the *IBM Tivoli Netcool/OMNIBus Installation and Deployment Guide*.

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

Loading the ObjectServer customization file

The ObjectServer customization file contains a configuration script that includes functionality for creating and populating conversions tables. The gateway provides an ObjectServer customization file called tivoli_eifgw_setup.sql.

Before running the gateway for the first time, you must load the tivoli_eifgw_setup.sql file. This SQL file ensures that the ObjectServer is configured with an appropriately populated conversions table to convert Tivoli Netcool/OMNIBus severity values to Tivoli Tivoli EIF severity values and creates the

resynchronization support table and triggers within the ObjectServer. It also adds the EIFFdBk column to the ObjectServer, which indicates when an event has been forwarded to Tivoli EIF.

To load the `tivoli_eifgw_setup.sql` file, execute the command for your operating system:

- On Linux and UNIX:

```
$OMNIHOME/bin/nco_sql -server objectserver -user username -password 'password' <
$OMNIHOME/gates/tivoli_eif/tivoli_eifgw_setup.sql
```

- On Windows:

```
%OMNIHOME%\bin\redist\isql.exe -U username -P 'password'
objectserver_name <
%OMNIHOME%\gates\tivoli_eif\tivoli_eifgw_setup.sql
```

Configuring the gateway to run with Tivoli EIF

The gateway can be configured to run with the Tivoli Event Integration Facility (Tivoli EIF) using the Tivoli EIF configuration file specified in the **Gate.TIVOLI{EIF.Config}** property, which is defined in the `NCO_GATE.props` file.

The following example shows sample settings for the configuration file specified in the **Gate.TIVOLI{EIF.Config}** property:

```
TransportList=t1
t1Type=SOCKET
t1Channels=c1
c1ServerLocation=tec server ip_address
c1Port=port_number
BufEvtPath=$OMNIHOME/var/tivoli_eif/eifcache
BufferEvents = YES
EventMaxSize=32768
```

For more information on Tivoli EIF, see *IBM Tivoli Netcool/OMNIBus Event Integration Facility Reference*.

Before running the gateway for the first time, you must make the following changes to the configuration file specified in the **Gate.TIVOLI{EIF.Config}** property:

1. Accept the default values for the **TransportList**, **t1Type**, and **t1Channels** properties.
2. Set the **c1ServerLocation** property to the IP address of the Tivoli EIF server.
3. Set the **c1Port** property to the port number of the Tivoli EIF server.

Note : On UNIX, the Tivoli EIF server uses a port mapper. To take advantage of this, set the port number to 0. On Windows, the Tivoli EIF server does not use a port mapper; the default port on Windows is 5529.

4. On UNIX and Linux operating systems, do not use the `$OMNIHOME` environment variable in the path specified in the **BufEvtPath** property. Instead, expand `$OMNIHOME` to the full path of the Tivoli Netcool/OMNIBus installation.

On Windows operating systems, do not use the `%OMNIHOME%` environment variable in the path specified in the **BufEvtPath** property. Instead, expand `%OMNIHOME%` to the full path of the Tivoli Netcool/OMNIBus installation.

5. Set the **BufferEvents** property to the value YES.
- 6.

Note : The **BufferEvents** and **BufEvtPath** properties determine store and forward (SAF) configuration. If you want to disable SAF, set the **BufferEvents** property to the value NO, in which case the **BufEvtPath** and **EventMaxSize** properties are not needed.

Replicating data using the 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 the following generic property:

Gate.RDRWTR.TblReplicateDefFile

The default table replication definition file is in the following directory: \$OMNIHOME/gates/tivoli_eif/tivoli_eif.rdrwtr.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 tables (alerts.status, alerts.journal, alerts.details) 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 6. Optional replication commands

Command	Description
FILTER WITH ' <i>filter</i> '	Filters the database rows selected for replication, where <i>filter</i> 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'
ORDER BY ' <i>order</i> '	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 <i>order</i> would be 'FirstOccurrence'.
AFTER IDUC DO ' <i>afteriduc</i> '	Updates replicated rows, where <i>afteriduc</i> 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'.

Optimizing the eif_gw_resync trigger

When many or most alerts within an ObjectServer are of no interest to the gateway and you use a filter in the `tivoli_eif.rdrwtr.tblrep.def` file to exclude the corresponding events, then you should modify the `eif_gw_resync` trigger within the `tivoli_eifgw_setup.sql` file to use the same gateway filter. This modification optimizes the `eif_gw_resync` trigger by reducing the number of ObjectServer alerts that are touched during resynchronization operations.

To modify the `eif_gw_resync` trigger within the `tivoli_eifgw_setup.sql` file, add a where clause that has the same effect as the filter that you are specifying for the gateway to the `update alerts.status` clause. For example, if you are using the filter `EIF_flag=1`, then extend the following sql statement:

```
update alerts.status set Severity = Severity;
```

to this:

```
update alerts.status set Severity = Severity where EIF_FLAG=1;
```

Mapping data using the map definition file

Use the **Gate.MapFile** property to specify the name and location of the file where the map is defined.

The table replication definition file (specified by the **Gate.RDRWTR.TblReplicationDefFile** property) names maps that are used in table replication and the **Gate.MapFile** property specifies the name and location of the file where the map is defined. The default location and name of the map definition file is as follows:

```
$OMNIHOME/gates/tivoli_eif/tivoli_eif.map
```

Note : In most cases you will not need to change the syntax of the map defined in the `tivoli_eif.map` map definition file.

For details related to the syntax of map definition files (including attribute names, cache value access attributes, dynamic attributes, and conversion functions), see the *IBM Tivoli Netcool/OMNIBus Probe and Gateway Guide*.

Attribute names

Attribute names enable you to include additional data in mapping definitions.

The following types of attributes can be specified:

- Cache value access attributes
- Dynamic attributes

This section describes the attribute names that can be used in the map definition file.

Cache value access attributes

Cache value access attributes enable the gateway to access values stored in the cross-reference cache.

The following table describes the cache value attributes that can be used in mapping definitions.

Table 7. Cache value access attributes	
Attribute name	Description
STATUS.SERIAL	Cached serial number for the status table row associated with the current journal or details table row.

Table 7. Cache value access attributes (continued)	
Attribute name	Description
STATUS.SERVER_SERIAL	Cached server serial number for the status table row associated with the current journal or details table row.
STATUS.SERVER_NAME	Cached server name for the status table row associated with the current journal or details table row.
STATUS.IDENTIFIER	Cached identifier for the status table row associated with the current journal or details table row.
JOURNAL.SERIAL	Cached serial number of the journal table row.
DETAILS.IDENTIFIER	Cached identifier of the details table row.

Dynamic attributes

Dynamic attributes enable the gateway to access dynamic values that are automatically generated by the gateway.

The following table describes the dynamic attributes that can be used in mapping definitions.

Table 8. Dynamic attributes	
Attribute name	Description
ACTION_CODE	Single character string that specifies the type of operation performed. Valid values are: I - Insert, U - Update, and D - Delete.
ACTION_TIME	Time in UTC that the action occurred.
DELETEDAT	If the action was a delete, this attributes displays the date at which the row was deleted.

Conversion functions

The conversion functions can be used in a map definition file.

The following conversion functions can be used in the map definition file:

- TO_STRING (*column_name*)
- TO_INTEGER (*column_name*)
- TO_TIME (*column_name*)

Example mapping

This example shows the mappings for the ObjectServer tables to which the gateway writes.

```
CREATE MAPPING StatusMap
(
  'identifier' = '@Identifier',
  'server_identifier' = '@ServerName' + " " +
  TO_STRING('@ServerSerial'),
  'sub_source' = '@AlertKey' ON INSERT ONLY,
  'sub_origin' = '@AlertGroup' ON INSERT ONLY,
  'msg' = '@Summary' ON INSERT ONLY,
  'origin' = '@Node' ON INSERT ONLY,
  'node_alias' = '@NodeAlias' ON INSERT ONLY NOTNULL
  '@Node',
  'manager' = '@Manager' ON INSERT ONLY,
```

```
'source' = '@Agent'ON INSERT ONLY,
'severity' = '@Severity',
'date' = '@LastOccurrence'ON INSERT ONLY,
'omnibus_last_modified_time' = '@InternalLast'ON INSERT ONLY );
);
```

Store and forward

If the Tivoli Event Integration Facility (Tivoli EIF) server goes offline, the Tivoli EIF API stores events in the cache file specified by the **BufEvtPath** property of the Tivoli EIF configuration file.

When the Tivoli EIF server is running again, the Tivoli EIF API forwards all events from the cache file to Tivoli EIF.

You specify the Tivoli EIF configuration file in the **Gate.TIVOLI{EIF.Config** property, which is defined in the `NCO_GATE.props` properties file. For details about configuring the Tivoli EIF configuration file, see [“Configuring the gateway to run with Tivoli EIF” on page 10](#).

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 for the gateway is called `tivoli_eif.startup.cmd` and is located in the following directory: `$OMNIHOME/gates/tivoli_eif`. Before making any edits to the `tivoli_eif.startup.cmd` file make a backup copy for future reference.

For information about the commands used in the startup command file, see the *IBM Tivoli Netcool/OMNIBus Probe and Gateway Guide*.

Note : 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*.

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
```

Gateway operations

Use this information to learn how the Gateway for Tivoli EIF performs a variety of operations.

For many of these gateway operations, you can control how the gateway executes these operations by setting specific properties defined in the `NCO_GATE.props` properties file.

The following topics describe how the Gateway for Tivoli EIF handles specific gateway operations:

- [“Understanding how the gateway handles resynchronization” on page 22](#)
- [“Understanding gateway error handling” on page 25](#)
- [“Understanding how the gateway controls buffer size” on page 25](#)
- [“Understanding how the gateway handles failback” on page 26](#)

Understanding gateway startup operations

The gateway provides configurable startup operations. Startup operations are provided by the IBM Tivoli Netcool/OMNIbus Gateway Toolkit (NGTK) library.

Specify the relevant properties in the gateway's EIF configuration file in the **Gate.TIVOLI{EIF.ConfigFile}** property defined in the `NCO_GATE.props` properties file.

Understanding gateway resynchronization operations

The gateway provides configurable resynchronization.

Specify the **Gate.TIVOLI{EIF.Resync}** and **Gate.TIVOLI{EIF.FullResync}** properties.

Identifying events sent by the gateway

The gateway adds a new Tivoli EIF attribute (*gateway_instance*) to all events (resync events and OMNIbus events) that it sends to the Tivoli EIF server.

The gateway populates the *gateway_instance* attribute with the value specified by the **Gate.TIVOLI{EIF.GWInstancename}** property defined in the `NCO_GATE.props` properties file. If you are running multiple gateways, you must use a different value for each Gateway for Tivoli EIF that you are running. This enables you to identify from which gateway each event originates.

Understanding event types

Use this information to understand the types of events that the gateway handles.

There are two types of events that the gateway handles:

- Resync events

Resync events provide information about resynchronization actions the gateway performs and the data required for resynchronization to be performed by the EIF target.

- OMNIbus events

OMNIbus events comprise event data where alerts are inserted, updated, and deleted in Tivoli Netcool/OMNIbus.

Understanding resync event formats

Use this information to understand the resync event formats that the gateway handles.

There are two types of resync event formats that the gateway handles:

- Full resync event format

On startup and when full resynchronization is enabled, the gateway sends a `full resync start` event to the EIF receiver. The `full resync start` event format consists of the following values:

Note : You enable full resynchronization by setting the **Gate.TIVOLI{EIF.Resync}** and **Gate.TIVOLI{EIF.FullResync}** properties to TRUE in the NCO_GATE.props properties file.

```
<Gate.TIVOLI{EIF.RsyncGWStartClass>;gateway_instance=  
<Gate.TIVOLI{EIF.GWInstanceName>
```

Where

- **Gate.TIVOLI{EIF.RsyncGWStartClass}** specifies a property defined in the NCO_GATE.props properties file. This property specifies the Tivoli EIF resync gateway start class name.
 - ; (semicolon) separates the property name from the *gateway_instance* value.
 - *gateway_instance* specifies the Tivoli EIF attribute that contains the name of the value specified in the **Gate.TIVOLI{EIF.GWInstanceName}** property defined in the NCO_GATE.props properties file.
 - **Gate.TIVOLI{EIF.GWInstanceName}** specifies the name of the gateway instance.
- Incremental resync event format (first event sent)

On startup and when incremental resynchronization is enabled, the gateway sends a number of events to the EIF receiver. The incremental event format for the first event sent consists of the following values:

```
<Gate.TIVOLI{EIF.RsyncDelStartClass>;gateway_instance=  
<Gate.TIVOLI{EIF.GWInstanceName>
```

Where

- **Gate.TIVOLI{EIF.RsyncDelStartClass}** specifies a property defined in the NCO_GATE.props properties file. This property specifies the Tivoli EIF resync gateway start delete class name.
- Incremental resync event format (subsequent events sent)

```
<Gate.TIVOLI{EIF.RsyncDelClass>;gateway_instance=  
<Gate.TIVOLI{EIF.GWInstanceName>;  
identifier_list=['<id1>', '<id2>', ... '<idn>']
```

Where

- *identifier_list* specifies 0 (zero) or more lists of Identifier values in single quotes, separated by commas.
- *id1*, *id2*, *idn* specify the Identifier values in this *identifier_list*.

In previous versions of the gateway, the size of each Identifier list is limited to 4096 bytes and so the number of Identifier lists depends on the sum of the lengths of Identifier values divided by the maximum list size. The current version of the gateway defines the

Gate.TIVOLI{EIF.MaxIDListSize} property that specifies the maximum size, in bytes, of each Identifier list sent during incremental resynchronization operations. The default maximum size is 4096 bytes.

- Incremental resync event format (final event sent)

The incremental event format for the first event sent consists of the following values:

```
<Gate.TIVOLI{EIF.RsyncDelEndClass>;gateway_instance=  
<Gate.TIVOLI{EIF.GWInstanceName>
```

Where

- **Gate.TIVOLI{EIF.RsyncDelEndClass}** specifies a property defined in the NCO_GATE.props properties file. This property specifies the Tivoli EIF resync gateway end delete class name.

Understanding how the gateway handles OMNibus events

Use this information to understand how the gateway handles OMNibus events.

Before the gateway forwards an alert that it has not seen before, an entry for the alert is created in the gateway's cache. The following list summarizes how the gateway handles OMNibus event inserts, updates, and deletes:

- The gateway drops OMNibus event inserts when the event Severity value is less than the value specified for the **Gate.TIVOLI{EIF.MinimumSeverity}** property defined in the `NCO_GATE.props` properties file. Note that the gateway logs the alert Serial.

After the gateway forwards the OMNibus event inserts to the EIF receiver, it updates the `EIFFdBk` field within the ObjectServer with a value of 1 to indicate that it has forwarded the OMNibus event to the EIF receiver.

- The gateway converts OMNibus event updates to OMNibus event inserts where they relate to alerts for which the gateway has not previously forwarded any OMNibus event.
- The gateway drops OMNibus event deletes if they relate to alerts not found in the gateway internal cache. That is, these OMNibus event deletes relate to an alert for which events have not been previously forwarded by the gateway. One likely reason for this situation is the original events were filtered out as a result of table replication file configuration.
- The gateway drops OMNibus event deletes if these events previously cleared (that is, they were updated with an event Severity value set to 0 (zero).

Understanding OMNibus event formats

Use this information to understand the OMNibus event formats that the gateway handles.

The OMNibus events that the gateway forwards to the EIF server are formatted as lists of name=value pairs defined in the map definition file. You specify the map definition file in the **Gate.MapFile** property defined in the `NCO_GATE.props` properties file. The default name of the mapping definition file is `tivoli_eif.map`. The name=value pairs are separated by semicolons (;) and values are enclosed within single quotes (for example, 'value1'). Each OMNibus event list begins with a prefix (using a property defined in the `NCO_GATE.props` properties file) and ends with a suffix (the keyword `END`). Target applications need to be configured to correctly parse these OMNibus events.

The OMNibus events can be classified as follows:

- OMNibus inserts events (identified by the **Gate.TIVOLI{EIF.InsertClass}** property)
- OMNibus updates events (identified by the **Gate.TIVOLI{EIF.UpdateClass}** property)
- OMNibus deletes events (identified by the **Gate.TIVOLI{EIF.DeleteClass}** property)

These property tags inform the target application the type of OMNibus event it is processing.

The following syntax description describes the format for each of these OMNibus events:

OMNibus inserts event format

```
<Gate.TIVOLI{EIF.InsertClass>; 1
gateway_instance=<Gate.TIVOLI{EIF.GWInstanceName>; 2
omnibus_status= 3
<value>;<name>;='<value>';...;
END
```

OMNibus updates event format

```
<Gate.TIVOLI{EIF.UpdateClass>; 1
gateway_instance=<Gate.TIVOLI{EIF.GWInstanceName>; 2
omnibus_status= 3
<??>;<name>;='<value>';...;
END
```

OMNibus deletes event format

```
<Gate.TIVOLI{EIF.DeleteClass>; 1
gateway_instance=<Gate.TIVOLI{EIF.GWInstanceName>; 2
omnibus_status= 3
<ServerSerial>;severity;=
```

```
'<previous Severity value>'
;END
```

The following numbered list describes each of the numbered items in the syntax description for the OMNIbus event formats:

1. Specify the property tags that identify each type of OMNIbus event. Each of these properties is defined in the `NCO_GATE.props` properties file. Note that a ; (semicolon) follows each property tag.
2. Specify the Tivoli EIF attribute that contains the name of the value specified in the **Gate.TIVOLI{EIF.GWInstanceName** property, which is defined in the `NCO_GATE.props` properties file. Note that a ; (semicolon) follows each **Gate.TIVOLI{EIF.GWInstanceName** property.
3. Specify one of the following values: `omnibus_status` takes one of the following values:
 - **ACKNOWLEDGED**
This value is specified when the alert's Acknowledged field is set to the value 1 (that is the alert is acknowledged) in Tivoli Netcool/OMNIbus.
 - **CLEARED**
This value is specified when the alert's Severity field is set to the value 0 (zero) or the event is deleted in Tivoli Netcool/OMNIbus.
 - **OPEN**
This value is specified for all other inserts and updates in Tivoli Netcool/OMNIbus.

Understanding OMNIbus event sizes

Use this information to understand OMNIbus event sizes.

In previous versions of the gateway, the size of OMNIbus events were limited to 4096 bytes. If an OMNIbus event exceeded 4096 bytes, the gateway failed. You can control the size of OMNIbus events by using the following properties defined in the `NCO_GATE.props` properties file:

- **Gate.TIVOLI{EIF.MaxIDListSize**

Use this property to specify the maximum size, in bytes, of each Identifier list sent during incremental resynchronization operations.

The default value is 4096 bytes. The maximum size is 16128 bytes. If you specify a value larger than 16128 bytes, the target application overrides the value.

- **Gate.TIVOLI{EIF.MaxEventSize**

Use this property to specify the maximum size, in bytes, of an OMNIbus event that the gateway will attempt to forward to the EIF receiver.

The default value is 4096 bytes. The recommended maximum size is 16128 bytes if EIF event buffering is on or 32512 bytes if EIF event buffering is off. If the gateway receives an event from Tivoli Netcool/OMNIbus that is larger than the value specified in the **Gate.TIVOLI{EIF.MaxEventSize** property, the gateway drops the event and logs the event contents, truncated at the maximum size. The gateway continues to run, but its execution depends on specific circumstances associated when EIF event buffering is enabled. For more information, see [“Understanding Tivoli EIF event buffering” on page 21](#).

The Identifier lists that Tivoli Netcool/OMNIbus sends during incremental resynchronization operations (as opposed to full resynchronization operations) inform the target application of all of the Identifiers currently within Tivoli Netcool/OMNIbus. This enables the target application to compare these with Identifiers it knows about. Any known Identifiers not included in the resynchronization lists must correspond to events that have been deleted from Tivoli Netcool/OMNIbus at a time when the gateway was unable to forward delete events. The total list of identifiers within Tivoli Netcool/OMNIbus is broken down into a set of lists, each of which is no larger in size than the value specified in the **Gate.TIVOLI{EIF.MaxIDListSize** property.

Understanding how the gateway handles events

Use this information to understand how the gateway handles events: both Tivoli Netcool/OMNIBus events and Tivoli EIF events.

The following topics describe how the gateway handles Tivoli Netcool/OMNIBus events and Tivoli EIF events.

Understanding Tivoli Netcool/OMNIBus event sizes

In previous versions of the gateway, the size of Tivoli Netcool/OMNIBus events were limited to 4096 bytes. If a Tivoli Netcool/OMNIBus event exceeded 4096 bytes, the gateway failed. You can control the size of Tivoli Netcool/OMNIBus events by using the following properties defined in the `NCO_GATE.props` properties file:

- **Gate.TIVOLI{EIF.MaxIDListSize**

Use this property to specify the maximum size, in bytes, of each Identifier list sent during incremental resynchronization operations.

The default value is 4096 bytes. The maximum size is 16128 bytes. If you specify a value larger than 16128 bytes, the target application overrides the value.

- **Gate.TIVOLI{EIF.MaxEventSize**

Use this property to specify the maximum size, in bytes, of a Tivoli Netcool/OMNIBus event that the gateway will attempt to forward to the EIF receiver.

The default value is 4096 bytes. The recommended maximum size is 16128 bytes if EIF event buffering is on or 32512 bytes if EIF event buffering is off. If the gateway receives an event from Tivoli Netcool/OMNIBus that is larger than the value specified in the **Gate.TIVOLI{EIF.MaxEventSize}** property, the gateway will drop the event and log the event contents, truncated at the maximum size. The gateway continues to run, but its execution depends on specific circumstances associated when EIF event buffering is on. For more information, see [“Understanding Tivoli EIF event buffering” on page 20](#).

The Identifier lists that Tivoli Netcool/OMNIBus sends during incremental resynchronization operations (as opposed to full resynchronization operations) inform the target application of all of the Identifiers currently within Tivoli Netcool/OMNIBus. This enables the target application to compare these with Identifiers it knows about. Any known Identifiers not included in the resynchronization lists must correspond to events that have been deleted from Tivoli Netcool/OMNIBus at a time when the gateway was unable to forward delete events. The total list of identifiers within Tivoli Netcool/OMNIBus is broken down into a set of lists, each of which is no larger in size than the value specified in the **Gate.TIVOLI{EIF.MaxIDListSize}** property.

Understanding the Tivoli EIF buffer cache

Tivoli EIF provides a store and forward (SAF) capability through the EIF buffer cache. You control whether Tivoli EIF store and forward (SAF) is on or off by setting the **BufferEvents** property to either YES (on) or NO (off) in the gateway's EIF configuration file. If set to YES (on), the gateway writes events to the EIF buffer cache. If set to NO (off), the gateway writes events directly to the EIF consumer using the EIF API.

Note : You specify the location and name of the gateway's EIF configuration file in the **Gate.TIVOLI{EIF.ConfigFile}** property defined in the `NCO_GATE.props` properties file.

You specify the name and location of the EIF buffer cache in the **BufEvtPath** property defined in the gateway's EIF configuration file. The default name of the EIF buffer cache file is `{EIF.cache}`.

You specify the size, in kilobytes (KB) of the EIF buffer cache in the **BufEvtMaxSize** property defined in the gateway's EIF configuration file. The default size of the EIF buffer cache file is 64 KB, which is likely to be inadequate.

When the EIF buffer cache file becomes full, it wraps around to the beginning, overwriting the oldest events with newer ones and so the old ones are lost.

Understanding Tivoli EIF event buffering

You control whether Tivoli EIF event buffering is on or off by setting the **BufferEvents** property to either YES (on) or NO (off) in the gateway's EIF configuration file. Note that the gateway is not aware of whether Tivoli EIF event buffering is on or off, only whether it is able to send events.

Note : You specify the location and name of the gateway's EIF configuration file in the **Gate.TIVOLI{EIF.ConfigFile}** property defined in the `NCO_GATE.props` properties file.

When Tivoli EIF event buffering is on (`BufferEvents=YES`), the gateway executes as follows:

- The gateway writes events to an EIF buffer cache whose location and name are specified in the **BufEvtPath** property in the gateway's EIF configuration file. For example, if the value for the **BufEvtPath** property is set to `$OMNIHOME/var/tec/eif.cache`, the gateway writes the events to the EIF buffer cache file named `eif.cache` in the `$OMNIHOME/var/tec` directory.
- If the EIF event consumer is not running when the gateway starts, the gateway continues running and writes events to the EIF cache for collection at a later time.
- The EIF event buffer cannot accommodate events that are larger than approximately 16128 bytes. If the **Gate.TIVOLI{EIF.MaxEventSize}** property is set to a value that exceeds 16128 bytes and an event larger than 16128 bytes is sent, the operation fails and the gateway shuts down.

Thus, the EIF event buffer provides a store and forward facility, whereby the gateway is able to send events while the EIF consumer is not running or able to consume them. The penalty for this functionality is a performance overhead, as events are written to and read from disk. Whether this overhead is detrimental or not depends on the event flow rate.

Note : When EIF event buffering is enabled, the gateway assumes that the EIF buffer cache is the target, not the ultimate EIF consumer. Thus, after the gateway successfully writes an event to the EIF buffer cache any subsequent errors (on the part of the gateway or the EIF API) will not be reported back to the gateway.

When Tivoli EIF event buffering is off (`BufferEvents=NO`), the gateway executes as follows:

- The EIF reads the events as the gateway writes them to the EIF buffer cache through the EIF API.
- The gateway will not start if the EIF event consumer is not already running.
- If the connection to the EIF receiver is lost, any attempt to forward an event fails and the gateway shuts down.
- If you set the **Gate.TIVOLI{EIF.MaxEventSize}** property to a value that exceeds 32512 bytes and an event that is oversized but within the limit is sent, this operation will fail and the gateway shuts down.

Note : If the gateway is unable to write events to the EIF buffer cache or if Tivoli EIF event buffering is off (`BufferEvents=NO`) and the gateway is unable to write to the EIF consumer (for example, because the EIF consumer is not running) then the gateway will shut down.

One drawback of using Tivoli EIF event buffering is the impact on performance in terms of event throughput: Tivoli EIF event buffering adds the overhead of writing events to the EIF buffer cache and reading them before forwarding them. Whether this is of any practical relevance depends on event volume. On the other hand, a drawback of *not* using Tivoli EIF event buffering is (should the target be unreachable) that the gateway shuts down and gets restarted repeatedly by the process agent (PA). If gateway resynchronization is enabled, there may be repeated updating and querying of the ObjectServer for Identifier list compilation.

Use of Tivoli EIF event buffering imposes a 16K event size limit, though this is unlikely to present a problem for most users of the gateway.

Understanding Tivoli EIF event buffering

Use this information to understand Tivoli EIF event buffering.

You control whether Tivoli EIF event buffering is on or off by setting the **BufferEvents** property to either YES (on) or NO (off) in the gateway's EIF configuration file. Note that the gateway is not aware of whether Tivoli EIF event buffering is on or off, only whether it is able to send events.

Note : You specify the location and name of the gateway's EIF configuration file in the **Gate.TIVOLI{EIF.ConfigFile}** property defined in the `NCO_GATE.props` properties file.

When Tivoli EIF event buffering is on (`BufferEvents=YES`), the gateway executes as follows:

- The gateway writes events to an EIF buffer cache whose location and name are specified in the **BufEvtPath** property in the gateway's EIF configuration file. For example, if the value for the **BufEvtPath** property is set to `$OMNIHOME/var/tec/eif.cache`, the gateway writes the events to the EIF buffer cache file named `eif.cache` in the `$OMNIHOME/var/tec` directory.
- If the EIF event consumer is not running when the gateway starts, the gateway continues running and writes events to the EIF cache for collection at a later time.
- The EIF event buffer cannot accommodate events that are larger than approximately 16128 bytes. If the **Gate.TIVOLI{EIF.MaxEventSize}** property is set to a value that exceeds 16128 bytes and an event larger than 16128 bytes is sent, the operation fails and the gateway shuts down.

Thus, the EIF event buffer provides a store and forward facility, whereby the gateway is able to send events while the EIF consumer is not running or able to consume them. The penalty for this functionality is a performance overhead, as events are written to and read from disk. Whether this overhead is detrimental or not depends on the event flow rate.

Note : When EIF event buffering is enabled, the gateway assumes that the EIF buffer cache is the target, not the ultimate EIF consumer. Thus, after the gateway successfully writes an event to the EIF buffer cache any subsequent errors (on the part of the gateway or the EIF API) will not be reported back to the gateway.

When Tivoli EIF event buffering is off (`BufferEvents=NO`), the gateway executes as follows:

- The EIF reads the events as the gateway writes them to the EIF buffer cache through the EIF API.
- The gateway will not start if the EIF event consumer is not already running.
- If the connection to the EIF receiver is lost, any attempt to forward an event fails and the gateway shuts down.
- If you set the **Gate.TIVOLI{EIF.MaxEventSize}** property to a value that exceeds 32512 bytes and an event that is oversized but within the limit is sent, this operation will fail and the gateway shuts down.

Note : If the gateway is unable to write events to the EIF buffer cache or if Tivoli EIF event buffering is off (`BufferEvents=NO`) and the gateway is unable to write to the EIF consumer (for example, because the EIF consumer is not running) then the gateway shuts down.

One drawback of using Tivoli EIF event buffering is the impact on performance in terms of event throughput: Tivoli EIF event buffering adds the overhead of writing events to the EIF buffer cache and reading them before forwarding them. Whether this is of any practical relevance depends on event volume. On the other hand, a drawback of *not* using Tivoli EIF event buffering is (should the target be unreachable) that the gateway shuts down and gets restarted repeatedly by the process agent (PA). If gateway resynchronization is enabled, there may be repeated updating and querying of the ObjectServer for Identifier list compilation.

Use of Tivoli EIF event buffering imposes a 16K event size limit, though this is unlikely to present a problem for most users of the gateway.

Understanding how the gateway handles resynchronization

Use the gateway properties defined in the `NCO_GATE.props` properties file to specify the type of resynchronization that the gateway performs with Tivoli EIF.

Table 9 on page 22 identifies the properties you use to specify the type of resynchronization the gateway performs with Tivoli EIF. The table also identifies properties related to the last state change value and events.

Table 9. Properties used in resynchronization	
Resynchronization property	Properties
Properties used to specify type of resynchronization	
Incremental resynchronization Gate.TIVOLI{EIF.Resync Gate.TIVOLI{EIF.FullResync	To instruct the gateway to perform incremental resynchronization with Tivoli EIF set the Gate.TIVOLI{EIF.Resync property to the value <code>TRUE</code> and the Gate.TIVOLI{EIF.FullResync property to the value <code>FALSE</code> . The gateway performs incremental resynchronization with Tivoli EIF by default.
Full resynchronization Gate.TIVOLI{EIF.Resync Gate.TIVOLI{EIF.FullResync	To instruct the gateway to perform full resynchronization with Tivoli EIF set the Gate.TIVOLI{EIF.Resync property to the value <code>TRUE</code> and the Gate.TIVOLI{EIF.FullResync property to the value <code>TRUE</code> .
No resynchronization Gate.TIVOLI{EIF.Resync	To instruct the gateway to perform no resynchronization with Tivoli EIF set the Gate.TIVOLI{EIF.Resync property to the value <code>FALSE</code> . Note : You would only specify that the gateway perform no resynchronization with Tivoli EIF in a test or experimental environment.
Property used to specify the name of the last state change file to which the gateway writes the last state change value during incremental resynchronization	
Gate.TIVOLI{EIF.LastStatechangeFile	When the gateway performs incremental resynchronization, it writes the last state change value to a file referred to as the last state change file. Use the Gate.TIVOLI{EIF.LastStatechangeFile property to specify the name (not the location) of the file that contains the last state change value. The default name of the last state change value file is <code>last_statechange.txt</code> .
Properties used to specify events that the gateway handles during incremental resynchronization	
First event sent Gate.TIVOLI{EIF.RsyncDelStartClass	Use this property to specify the Tivoli EIF resync delete start class name. The default is <code>OMNIBus_Delete_Start</code> .

Table 9. Properties used in resynchronization (continued)	
Resynchronization property	Properties
Subsequent events sent Gate.TIVOLI_EIF.RsyncDelClass	Use this property to specify the Tivoli EIF resync delete class name. The default is OMNIBus_Delete.
Final event sent Gate.TIVOLI_EIF.RsyncDelEndClass	Use this property to specify the Tivoli EIF resync delete end class name. The default is OMNIBus_Delete_End.
Properties used to specify events that the gateway handles during full resynchronization	
Gateway start Gate.TIVOLI_EIF.RsyncGWStartClass	Use this property to specify the Tivoli EIF resync gateway start class name. The default is OMNIBus_Gateway_Start.
Properties used to specify Tivoli Netcool/OMNIBus events that the gateway forwards to the Tivoli EIF server	
OMNIBus inserts events Gate.TIVOLI_EIF.InsertClass	Use this property to specify the Tivoli EIF insert event class name. The default is OMNIBus.
OMNIBus updates events Gate.TIVOLI_EIF.UpdateClass	Use this property to specify the Tivoli EIF update class name. The default is OMNIBus_Update.
OMNIBus deletes events Gate.TIVOLI_EIF.DeleteClass	Use this property to specify the Tivoli EIF delete event class name. The default is OMNIBus_Update.

Understanding how the gateway handles full resynchronization

Use this information to understand how the gateway handles full resynchronization.

On startup, the gateway performs a full resynchronization with EIF when the **Gate.TIVOLI_EIF.Resync** and **Gate.TIVOLI_EIF.FullResync** properties defined in the NCO_GATE.props properties file are set to the values TRUE.

The gateway sends a `full resync start` resync event to EIF to signify that a full resynchronization is commencing. This provides the EIF consumer with an opportunity to perform any user-required actions in the target. For example, the user-required action might be to purge any data previously sent by the gateway.

The data that the `eif_gw_resync trigger` writes to the `alerts.eif_gw_resync` table in Tivoli Netcool/OMNIBus causes the `EIFFdBk` field in the `alerts.status` table for all alerts to be set to the value 0 (zero). This effectively marks all alerts in Tivoli Netcool/OMNIBus as having never been seen before by the gateway and by extension the EIF receiver. The `eif_gw_resync trigger` updates all alerts to cause them to be forwarded to the gateway through the normal IDUC notification process.

Understanding how the gateway handles incremental resynchronization

Use this information to understand how the gateway handles incremental resynchronization.

On startup, the gateway performs an incremental resynchronization with EIF when the **Gate.TIVOLI{EIF}.Resync** and **Gate.TIVOLI{EIF}.FullResync** properties defined in the `NCO_GATE.props` properties file are set to the values `TRUE` and `FALSE`, respectively.

Incremental resynchronization makes use of the `alerts.eif_gw_resync` table in Tivoli Netcool/OMNIbus and an associated trigger called `eif_gw_resync` trigger. The `tivoli_eifgw_setup.sql` script installs both the `alerts.eif_gw_resync` table and the `eif_gw_resync` trigger. The `tivoli_eifgw_setup.sql` script delivers with the gateway and needs to be loaded into the ObjectServer using the `nco_sql` utility.

During incremental resynchronization, the gateway performs the following operations:

- First, the gateway informs the EIF receiver of all of the Identifier values that exist in the `alerts.status` table in the ObjectServer. This gives the EIF target the opportunity to compare this list with Identifiers that it knows about. Any that it knows about that are missing from the ObjectServer can be inferred as having been deleted from the ObjectServer before the gateway started. See [“Understanding resync event formats”](#) on page 15 for details about the resync event formats. of the resync start, identifier list(s) and resync end messages that are sent.
- Secondly, with the aid of the `eif_gw_resync` trigger, the gateway causes alerts that have been inserted or updated since the gateway's last run to be updated. To accomplish this, the gateway reads the last statechange value written to the last statechange file on startup to determine the latest statechange of any event the gateway has seen coupled with the writing of data to the `alerts.eif_gw_resync` table. Recent unseen versions of alerts are then updated by the `eif_gw_resync` trigger to cause them to be forwarded to the gateway through the normal IDUC notification process.

In previous versions of the gateway, the file containing the last statechange value (if any) is `$NCHOME/var/tivoli_eif/last_statechange.txt` and is not configurable. Thus, if more than one gateway runs within a Tivoli Netcool/OMNIbus installation, a conflict occurs.

The current version of the gateway defines a property called

Gate.TIVOLI{EIF}.LastStatechangeFile in the `NCO_GATE.props` properties file. With this property, you can specify the name (not the location) of the last state change file.

See [“Understanding how the gateway makes use of the latest state change file”](#) on page 24 for information about the last state change file and the **Gate.TIVOLI{EIF}.LastStatechangeFile** property.

Understanding how the gateway makes use of the latest state change file

Use this information to understand how the gateway makes use of the latest state change file during incremental resynchronization.

To aid incremental resynchronization on startup, the gateway writes the latest statechange value of any ObjectServer event seen within an Insert, Delete, Update, or Control (IDUC) batch of ObjectServer events to a file when it has finished processing the batch. This file is referred to as the latest state change file. You specify the name of the latest state change file using the

Gate.TIVOLI{EIF}.LastStatechangeFile property defined in the `NCO_GATE.props` properties file. The default name of the file that contains the last state change value is `last_statechange.txt`.

Note : The directory that contains the `last_statechange.txt` file is `$OMNIHOME/var/`
Gate.TIVOLI{EIF}.GWInstanceName *string*. The **Gate.TIVOLI{EIF}.GWInstanceName** property is defined in the `NCO_GATE.props` properties file.

On startup, the ObjectServer reads the latest statechange file and touches all Tivoli Netcool/OMNIbus events that have changed at a later time than the last statechange value. This action causes the gateway to retrieve the touched Tivoli Netcool/OMNIbus events, potentially for forwarding to the EIF consumer.

If the gateway does not forward a Tivoli Netcool/OMNIbus within a batch, the last statechange value is not updated. If a Tivoli Netcool/OMNIbus event is dropped because it is too large, the statechange value

will be updated as normal at the end of the processing of the batch. If the gateway shuts down because of an error in forwarding an event to EIF, the last statechange value will not be updated. Thus when the gateway restarts, resynchronization causes the last batch to be resent. This may include some events that were previously forwarded as well as those that were not.

If no statechange file exists when the gateway starts then all Tivoli Netcool/OMNIBus events will be touched and retrieved by the gateway, as is the case with full resynchronization, and no Identifier list will be sent.

Understanding gateway error handling

The gateway provides configurable error handling. Error handling is provided by the IBM Tivoli Netcool/OMNIBus Gateway Toolkit (NGTK) library.

To specify that the NGTK library logs debug messages, set the **Gate.NGtkDebug** property to TRUE.

You can specify which debug messages are included in the debug files by using the **Gate.Mapper.Debug** and **Gate.RDRWTR.Debug** properties; these can be set to TRUE or FALSE as appropriate.

Understanding how the gateway runs under process agent control

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

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

These properties are maintained automatically by the PA server and provide information only. They should not be changed manually.

Understanding the Tivoli EIF buffer cache

Use this information to understand the Tivoli EIF buffer cache.

Tivoli EIF provides a store and forward (SAF) capability through the EIF buffer cache. You control whether Tivoli EIF store and forward (SAF) is on or off by setting the **BufferEvents** property to either YES (on) or NO (off) in the gateway's EIF configuration file. If set to YES (on), the gateway writes events to the EIF buffer cache. If set to NO (off), the gateway writes events directly to the EIF consumer using the EIF API.

Note : You specify the location and name of the gateway's EIF configuration file in the **Gate.TIVOLI{EIF.ConfigFile}** property defined in the NCO_GATE . props properties file.

You specify the name and location of the EIF buffer cache in the **BufEvtPath** property defined in the gateway's EIF configuration file. The default name of the EIF buffer cache file is `EIF.cache`.

You specify the size, in kilobytes (KB) of the EIF buffer cache in the **BufEvtMaxSize** property defined in the gateway's EIF configuration file. The default size of the EIF buffer cache file is 64 KB, which is likely to be inadequate.

When the EIF buffer cache file becomes full, it wraps around to the beginning, overwriting the oldest events with newer ones and so the old ones are lost.

Understanding how the gateway controls buffer size

The buffer size controls the number of entries that the gateway stores in its buffer before flushing them to the Tivoli EIF server.

This can be set using the **Gate.RDRWTR.BufferSize** property. This property can be adjusted to fine-tune the efficiency of the gateway.

The optimum value for the buffer size depends upon the average event size and the speed of the network. The default value has proved to be efficient for many installations. To determine the most efficient setting for your system, compare the timing figures for resynchronization operations performed using different settings for this property.

Understanding how the gateway handles failback

Failback functionality comes into operation when a gateway loses its connection to the primary ObjectServer. The failback functionality enables the gateway to connect to a backup ObjectServer.

Failback functionality also enables the gateway to reconnect to the primary ObjectServer when it becomes active again.

Setting up failback

To set up failback, set the ObjectServer **BackupObjectServer** property for the backup ObjectServer to TRUE. For details about setting ObjectServer properties, see the *IBM Tivoli Netcool/OMNIBus Administration Guide*.

To enable failback, in the gateway properties file, set the **Gate.RDRWTR.FailbackEnabled** property to TRUE. When the primary ObjectServer fails, the reader/writer fails over to the backup ObjectServer without shutting down. When the reader/writer detects that it is now connected to a backup ObjectServer, it periodically polls for the return of the primary ObjectServer. When the primary ObjectServer has been detected again, the reader/writer automatically fails back to the primary ObjectServer.

To specify the frequency with which the reader/writer part of the gateway polls the failed (primary) ObjectServer, set the **Gate.RDRWTR.FailbackTimeout** property.

Running the Gateway

The Gateway for Tivoli EIF can run in a variety of ways. The way you chose depends on a number of factors, including your operating system or your environment.

The following topics describe how to run the Gateway for Tivoli EIF:

- [“Running the Gateway on UNIX” on page 26](#)
- [“Running the Gateway on Windows” on page 26](#)

Running the Gateway on UNIX

The Gateway for Tivoli EIF can run on a UNIX operating system.

To run the gateway on UNIX, enter the following command on the command line:

```
$OMNIHOME/bin/nco_g_tivoli_eif
```

Note : To reduce latency, run the gateway on the same machine to which it writes alerts. You can also reduce latency by adjusting the buffer size. For details, see [“Understanding how the gateway controls buffer size” on page 25](#).

Running the Gateway on Windows

The Gateway for Tivoli EIF can be run on the Windows operating system and it can also run as a Windows service.

To run the gateway on Windows, run the following command:

```
%\OMNIHOME%\bin\nco_g_tivoli_eif.exe
```

or use the `-config` command line option as follows:

```
%\OMNIHOME%\bin\nco_g_tivoli_eif.exe -config \path\config_filename
```

Where *path* is the location of the configuration file and *config_filename* is the name of the configuration file.

Running the Gateway as a Windows Service

To run the gateway as a Windows service, use the following steps:

1. Run the following command to register the gateway as a service within the Service Control Manager:

```
nco_g_tivoli_eif.exe -install
```

2. If you are running the gateway on the same machine as the ObjectServer, use the following command to register its dependency upon the ObjectServer service:

```
nco_g_tivoli_eif.exe -install -depend NCOObjectServer
```

3. Select the following option from the **Start** menu: .

Control Panel > Administrative Tools > Services

The Services window is displayed.

4. Highlight the Tivoli EIF Gateway in the list of available services.
5. Click on **Start**.

Removing the Gateway Service

To remove the gateway service, use the following steps:

1. Stop the service using the Services window.
2. Run the following command:

```
nco_g_tivoli_eif.exe -remove
```

Properties and command line options

The properties file specifies how the gateway interacts with the device. You can override the default values by changing the default values in the properties file or by using the command line options. The properties file for the Gateway for Tivoli EIF is NCO_GATE.props.

Table 10 on page 27 describes the properties and command line options for the properties defined in the NCO_GATE.props properties file.

Note : For information about the common Tivoli Netcool/OMNIbus properties, see the *IBM Tivoli Netcool/OMNIbus Probe and Gateway Guide*.

For information about the Interprocess Communication (IPC) properties, see the *IBM Tivoli Netcool/OMNIbus Probe and Gateway Guide*.

Table 10. Properties and command line options		
Property name	Command line option	Description
Gate.CacheHashTblSize <i>integer</i>	-chashtblsize <i>integer</i>	Use this property to specify the number of elements that the gateway allocates for the hash table cache. The default is 5023 elements.

Table 10. Properties and command line options (continued)

Property name	Command line option	Description
Gate.MapFile <i>string</i>	-mapfile <i>string</i>	Use this property to specify the location and name of the map definition file. For details of the format of this file, see “Mapping data using the map definition file” on page 12. On UNIX and Linux operating systems, the default is \$OMNIHOME/gates/tivoli_eif/tivoli_eif.map. On Windows operating systems, the default is %OMNIHOME%\gates\tivoli_eif\tivoli_eif.map.
Gate.NGtkDebug <i>boolean</i>	-ngtkdebug <i>boolean</i>	Use this property to specify whether the NGTK library should log debug messages. The default is TRUE. Note : You can specify which debug messages are included in the debug log file using the Gate.Mapper.Debug and Gate.RDRWTR.Debug properties.
Gate.PAAware <i>integer</i>	-paaware <i>integer</i>	See the <i>IBM Tivoli Netcool/OMNIBus Probe and Gateway Guide</i> for information on this property.
Gate.PAAwareName <i>string</i>	-paname <i>string</i>	See the <i>IBM Tivoli Netcool/OMNIBus Probe and Gateway Guide</i> for information on this property.
Gate.StartupCmd File <i>string</i>	-startupcmdfile <i>string</i>	Use this property to specify the location and name of the startup command file. For details of the format of the this file, see “Startup command file” on page 14. On UNIX and Linux operating systems, the default is \$OMNIHOME/gates/tivoli_eif/tivoli_eif.startup.cmd. On Windows operating systems, the default is %OMNIHOME%\gates\tivoli_eif\tivoli_eif.startup.cmd.
Gate.Transfer.FailoverSyncRate <i>string</i>	-transferfailover syncrate <i>string</i>	Use this property to specify the rate (in seconds) at which the re-synchronization takes place on failover. The default is 60 seconds.

Table 10. Properties and command line options (continued)

Property name	Command line option	Description
Gate.RDRWTR.BufferSize <i>integer</i>	<code>-rdwtrbufsize</code> <i>integer</i>	Use this property to specify the number of entries that the gateway stores in the buffer before flushing, if buffering is enabled. This property can be used to fine-tune the efficiency of the gateway. For details, see “Understanding how the gateway controls buffer size” on page 25. The default is 25. Note : The gateway flushes the buffer when the end of a batch of SQL statements has been reached regardless of the buffer size.
Gate.RDRWTR.Debug <i>boolean</i>	<code>-rdwtrdebug</code> <i>boolean</i>	Use this property to specify whether the gateway includes gateway reader debug messages in the debug log. The default is TRUE.
Gate.RDRWTR.DeleteIfNoDedup <i>string</i>	<code>-rdwtrdeleteifnodup</code> <i>string</i>	Use this property to specify whether the Tivoli EIF deletes duplicate events. The default is FALSE.
Gate.RDRWTR.Description <i>string</i>	<code>-rdwtrdescription</code> <i>string</i>	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/Writer.
Gate.RDRWTR.FailbackEnabled <i>boolean</i>	<code>-rdwtrfailbackenabled</code> <i>boolean</i>	Use this property to enable failback to a primary system. For details about failback functionality, see “Understanding how the gateway handles failback” on page 26. The default is FALSE.
Gate.RDRWTR.FailbackTimeout <i>integer</i>	<code>-rdwtrfailbacktimeout</code> <i>integer</i>	Use this property to specify the time (in seconds) that the gateway allows before entering failback mode. The default is 30 seconds.

Table 10. Properties and command line options (continued)

Property name	Command line option	Description
Gate.RDRWTR.IducFlushRate <i>integer</i>	<code>-rdwtriducflushrate</code> <i>integer</i>	Use this property to specify the rate (in seconds) of the granularity of the reader. If you set this property to 0, the reader gets its updates at the same granular rate as that of the ObjectServer to which it is connected. 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 this frequency. This enables the reader to run at a faster granularity than that of the ObjectServer, thus enabling the gateway to capture more detailed event changes in systems where the ObjectServer itself has high granularity settings.
Gate.RDRWTR.LogOSSql <i>boolean</i>	<code>-rdwtrlogossql</code> <i>boolean</i>	Use this property to specify whether the gateway logs all SQL commands sent to the ObjectServer in debug mode. The default is FALSE.
Gate.RDRWTR.Password <i>string</i>	<code>-rdwtrpassword</code> <i>string</i>	Use this property to specify the password associated with the user specified by the Gate.RDRWTR.Username property. The default is "". If the ObjectServer from which the gateway reads alerts is running on Tivoli Netcool/OMNIBus V7.2, or V7.2.1 this password must be encrypted by the <code>nco_g_crypt</code> utility. If the ObjectServer from which the gateway reads alerts is running on Tivoli Netcool/OMNIBus V7.2.1 in FIPS 140-2 mode, this password must be either plain text or encrypted using the <code>nco_aes_crypt</code> utility. Note : For details about the <code>nco_g_crypt</code> and <code>nco_aes_crypt</code> encryption utilities, see the <i>IBM Tivoli Netcool/OMNIBus Administration Guide</i>.

Table 10. Properties and command line options (continued)

Property name	Command line option	Description
Gate.RDRWTR.ReconnectTimeout <i>integer</i>	<code>-rdwtrreconntimeout <i>integer</i></code>	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 seconds.
Gate.RDRWTR.RefreshCacheOnUpdate <i>integer</i>	<code>-rdwtrrefreshcacheonupdate <i>integer</i></code>	Use this property to specify whether the Tivoli EIF refreshes cache on each update. The default is FALSE.
Gate.RDRWTR.Server <i>string</i>	<code>-rdwtrserver <i>string</i></code>	Use this property to specify the name of the ObjectServer from which the gateway reads alerts The default is NCOMS.
Gate.RDRWTR.StatusTableName <i>string</i>	<code>-rdwtrstatustbl <i>name string</i></code>	Use this property to specify the name of the status table that the gateway reads. The default is <code>alerts.status</code> .
Gate.RDRWTR.TblReplicateDefFile <i>string</i>	<code>-rdwtrtblrepdef <i>file string</i></code>	Use this property to specify the location and name of the table replication definition file. On UNIX and Linux operating systems, the default is <code>\$OMNIHOME/gates/tivoli_eif/tivoli_eif.rdwtr.tblrep.def</code> . On Windows operating systems, the default is <code>%OMNIHOME%\gates\tivoli_eif\tivoli_eif.rdwtr.tblrep.def</code> .
Gate.RDRWTR.UseBulkInsCmd <i>boolean</i>	<code>-rdwtrnobulkinscmd <i>boolean</i></code>	Use this property to specify whether the gateway uses the bulk insert SQL command. The default is FALSE.
Gate.RDRWTR.UserName <i>string</i>	<code>-rdwtrusername <i>string</i></code>	Use this property to specify the username used to authenticate the ObjectServer connection. The default is <code>root</code> .

Table 10. Properties and command line options (continued)

Property name	Command line option	Description
Gate.TIVOLI_EIF.ConfigFile <i>string</i>	<code>-tivoli_eifconfig</code> <i>string</i>	Use this property to specify the location and name of the Tivoli EIF configuration file. On UNIX and Linux operating systems, the default is \$OMNIHOME/gates/tivoli_eif/tivoli_eif_config. On Windows operating systems, the default is %OMNIHOME%\gates\tivoli_eif\tivoli_eif_config.
Gate.TIVOLI_EIF.DeleteClass <i>string</i>	<code>-tivoli_eifdelclass</code> <i>string</i>	Use this property to specify the Tivoli EIF delete event class name. The default is OMNIbus_Update.
Gate.TIVOLI_EIF.FullResync <i>boolean</i>	<code>-tivoli_eiffullresync</code> <i>boolean</i>	Use this property to specify whether the gateway performs a full resynchronization on startup. The default is FALSE.
Gate.TIVOLI_EIF.GWInstanceName <i>string</i>	<code>-tivoli_eifinstname</code> <i>string</i>	Use this property to specify the name of the gateway instance. The default is EIFGW. Note : You must use a different value for each Tivoli EIF gateway that you are running. This value is used for resynchronization to allow Tivoli EIF to identify from which gateway each event originates.
Gate.TIVOLI_EIF.InsertClass <i>string</i>	<code>-tivoli_eifinsclass</code> <i>string</i>	Use this property to specify the Tivoli EIF insert event class name. The default is OMNIbus.
Gate.TIVOLI_EIF.LastState changeFile <i>string</i>	<code>-tivoli_eiflaststate</code> <code>change</code> <i>string</i>	Use this property to specify the name (not the location) of the file that contains the last state change value. The default name of the file that contains the last state change value is last_statechange.txt. Note : The directory that contains the last_statechange.txt file is \$OMNIHOME/var/ Gate.TIVOLI_EIF.GWInstanceName <i>string</i> .

Table 10. Properties and command line options (continued)

Property name	Command line option	Description
Gate.TIVOLI_EIF. MaxEventSize <i>integer</i>	-tivoli_eifmax eventsize <i>integer</i>	Use this property to specify the maximum size, in bytes, of a Tivoli Netcool/OMNIBus event that the gateway will attempt to forward to the EIF receiver. The default value is 4096 bytes. The recommended maximum size is 16128 bytes if EIF event buffering is configured or 32512 bytes if EIF event buffering is not configured.
Gate.TIVOLI_EIF. MaxIDListSize <i>integer</i>	-tivoli_eifmax idlistsize <i>integer</i>	Use this property to specify the maximum size, in bytes, of each Identifier list sent during incremental resynchronization operations. The default value is 4096 bytes. The maximum size is 16128 bytes.
Gate.TIVOLI_EIF. MinimumSeverity <i>integer</i>	-tivoli_eifminsev idlistsize <i>integer</i>	Use this property to pass inserts only for alerts whose severity is greater than or equal to the value specified in this property. The default value is 3 (Minor).
Gate.TIVOLI_EIF. ReconnectTimeout <i>integer</i>	-tivoli_eifre conntimeout <i>integer</i>	Use this property to specify the time (in seconds) between each reconnection poll attempt if the connection to Tivoli EIF is lost. The default is 30 seconds.
Gate.TIVOLI_EIF. Resync <i>boolean</i>	-tivoli_eif noresync <i>boolean</i>	Use this property to specify whether the gateway performs resynchronization on startup. The default is TRUE (the gateway performs resynchronization on startup).
Gate.TIVOLI_EIF. RsyncDelClass <i>string</i>	-tivoli_eifrsyncdel class <i>string</i>	Use this property to specify the Tivoli EIF resync delete class name. The default is OMNIBus_Delete.
Gate.TIVOLI_EIF. RsyncDelEndClass <i>string</i>	-tivoli_eifrsync delendclass <i>string</i>	Use this property to specify the Tivoli EIF resync delete end class name. The default is OMNIBus_Delete_End.
Gate.TIVOLI_EIF.R syncDelStartClass <i>string</i>	-tivoli_eifrsyncdel startclass <i>string</i>	Use this property to specify the Tivoli EIF resync delete start class name. The default is OMNIBus_Delete_Start.

Table 10. Properties and command line options (continued)

Property name	Command line option	Description
Gate.TIVOLI{EIF.RsyncGWStartClass <i>string</i>	<code>-tivoli_eifrsync startclass <i>string</i></code>	Use this property to specify the Tivoli EIF resync gateway start class name. The default is OMNIBus_Gateway_Start.
Gate.TIVOLI{EIF.Simulator <i>boolean</i>	<code>-tivoli_eif simulator <i>boolean</i></code>	Use this property to specify whether the gateway writes events to a file instead of to Tivoli EIF. The default is FALSE. Note : If this property is set to TRUE, the gateway writes events to the file <code>tivoli_eif_simulator.txt</code> .
Gate.TIVOLI{EIF.UpdateClass <i>boolean</i>	<code>-tivoli_eif updateclass <i>boolean</i></code>	Use this property to specify the Tivoli EIF update class name. The default is OMNIBus_Update.
N/A	<code>-utf8enabled <i>string</i></code>	Use this command line option to control the encoding of data that is passed into, or generated by, the Windows version of the probe. Set this command line option to TRUE to run the probe in UTF-8 mode. The default is FALSE (which instructs the probe to use the default system code page).

Known issues

This section contains information about Known issues.

Dropping lower severity events

By default, the gateway drops events that have a severity of 0, 1, and 2. You can change this by using the **Gate.TIVOLI{EIF.MinimumSeverity** property.

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.



SC23-9117-07

